UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DISSERTAÇÃO DE MESTRADO

UMA ANÁLISE EMPÍRICA DE ABORDAGENS PARA A GERAÇÃO DE TESTES ABSTRATOS BASEADOS EM MODELOS DE REDES DE PETRI COLORIDAS

MESTRANDA
IALLY CRISTINA SILVEIRA DE ALMEIDA

ORIENTADOR
PROF. DR. LEANDRO DIAS DA SILVA

CO-ORIENTADOR
PROF. DR. ÁLVARO ALVARES DE CARVALHO CÉSAR SOBRINHO

MACEIÓ, AL
AGOSTO - 2020

Ially Cristina Silveira de Almeida

Orientador(a)

Leandro Dias da Silva

Coorientador(a)

Álvaro Alvares de Carvalho César Sobrinho

Maceió, AL

Agosto - 2020

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico

Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 - 1767

A447a Almeida, Ially Cristina Silveira de.
Uma análise empírica de abordagens para a geração de testes abstratos baseados em modelos de redes Petri coloridas / Ially Cristina Silveira de Almeida. – 2020.
74 f. : il.

Orientador: Leandro Dias da Silva.
Co-orientador: Álvaro Álvares de Carvalho César Sobrinho.
Dissertação (mestrado em Informática) - Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2020.

Bibliografia: f. 67-73.
Anexo: f. 74.

1. Tipos abstratos de dados (Computação). 2. Petri, Redes de. 3. Tradução de linguagem de programação. I. Título.

CDU: 004.652.3

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL
Programa de Pós-Graduação em Informática – PpgI
Instituto de Computação
Campus A. C. Simões BR 104-Norte Km 14 BL 12 Tabuleiro do Martins
Maceió/AL - Brasil CEP: 57.072-970 | Telefone: (082) 3214-1401

Folha de Aprovação

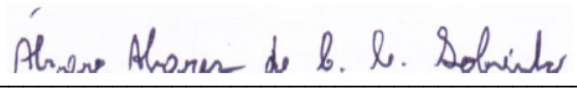
Ially Cristina Silveira de Almeida

"UMA ANÁLISE EMPÍRICA DE ABORDAGENS PARA A GERAÇÃO DE TESTES ABSTRATOS BASEADOS EM MODELOS DE REDES DE PETRI COLORIDAS"

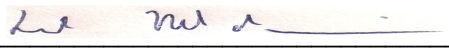
Dissertação submetida ao corpo docente do Programa de Pós-Graduação em Informática da Universidade Federal de Alagoas e aprovada em 28 de agosto de 2020.

Banca Examinadora:

Prof. Dr. Leandro Dias da Silva
UFAL – Instituto de Computação
Orientador



Prof. Dr. Alvaro Alvares de Carvalho Cesar Sobrinho
UFAPE – Universidade Federal do Agreste de Pernambuco
Coorientador



Prof. Dr. Leonardo Melo de Medeiros
IFAL-Instituto Federal de Alagoas
Examinador Interno

Prof. Dr. Angelo Perkusich
UFCG-Universidade Federal de Campina Grande
Examinador Externo

Prof. Dr. Lenardo Chaves e Silva
UFERSA - Universidade Federal Rural do Semi-Árido
Examinador Externo

Agradecimentos

Primeiramente, agradeço a Deus por estar comigo nessa caminhada, me dando forças e guiando meu caminho nos dias felizes e também nos de dificuldade. Agradeco aos meus pais Izabel e Cláudio, e aos meus irmãos, Cláudio Jr. e Haylla, que sempre acreditaram em mim, mesmo quando eu parei de acreditar. A força e a coragem que vinham de suas palavras e seu apoio nos momentos difíceis me trouxeram até aqui e para isso eu não tenho palavras suficientes para agradecer.

Agradeço ao meu agora marido, Jonathas Alberto. Sem ele, eu não estaria aqui. Depois de muitas tentativas em entrar num programa de mestrado, eu já havia desistido. Sua insistência em pedir que eu tentasse só mais uma vez me trouxe até esse momento. Foram dias e noites de muita luta emocional e em todas elas você esteve presente, ouvindo, aconselhando ou só estando comigo. Eu te amo mais do que posso expressar nessas poucas linhas.

Aos meus orientadores, Professor Leandro Dias e Professor Álvaro Sobrinho, por todo apoio, direcionamento, ajuda, conselhos e motivação. Em alguns momentos me senti perdida, e vocês conseguiram me ajudar a encontrar meu caminho novamente. Muito obrigada por tudo!

Aos amigos que fiz do mestrado, Andressa e João, que se tornaram amigos de vida, Adriano, meu colega de laboratório, que muito me ajudou e incentivou no meu trabalho, Anderson, da secretaria do PPGI que sempre estava disposto a me ouvir e me ajudar e a todos os colegas que fiz. A caminhada foi mais leve com vocês. Essa é uma realização conjunta, de muitas pessoas que me ajudaram a chegar aqui, e a todas elas eu agradeço de coração.

Agradeço também à CAPES pelo apoio financeiro durante a pesquisa.

Gostaria de dedicar esse trabalho à minha sobrinha Cecília. Assim como eu, ela tem uma forma diferente de lidar com o mundo, e nem sempre as pessoas irão entendê-la. Quero que quando ela cresça, eu possa contar pra ela a história desses dois anos da minha vida e de como não devemos desistir de nada que nos propusemos a fazer. Espero que a história a motive e torço pra que ela cresça e aprenda a lidar com os desafios da vida e do mundo com a certeza de que dias ruins sempre existirão. Mas os dias melhores chegam logo após, e por eles, tudo vale a pena.

Resumo

O teste baseado em modelos (*Model-Based Testing - MBT*) utiliza modelos do comportamento do sistema para gerar testes abstratos. Testadores reutilizam especificações formais (*e.g.*, modelos de redes de Petri coloridas (*Coloured Petri Nets - CPN)*) para projetar testes para sistemas críticos seguros. Neste trabalho, por uma revisão terciária, foi identificado um número considerável de revisões de literatura focadas na análise do uso de linguagens de especificação para realizar MBT. Entretanto, ainda existe uma lacuna de pesquisa em relação à análise de abordagens baseadas em CPN para geração de testes abstratos. Para preencher essa lacuna de pesquisa, neste trabalho, uma análise empírica de abordagens para geração de testes abstratos com base em modelos CPN foi também realizada, por meio de uma revisão sistemática da literatura e um estudo de caso sobre sistemas médicos: eletrocardiografia e bomba de infusão de insulina. A partir da análise empírica, são fornecidas informações para os testadores que precisam selecionar a abordagem de geração de testes abstratos mais adequada ao aplicar MBT usando CPN. Com os resultados obtidos, é possível identificar que a escolha depende do tamanho do espaço de estados do modelo CPN.

Palavras-chave: Testes baseados em modelos, testes abstratos, especificações formais, redes de Petri.

Abstract

The model-based testing (MBT) relies on models of the system's behavior to generate test sequences. Testers usually reuse formal models designed using specification languages such as coloured Petri nets (CPN) and state machines to design tests for safety-critical systems. There is a considerable number of research studies focused on analyzing the usage of specification languages to conduct MBT. However, there is still a research gap regarding the analysis of test sequence generation approaches designed for CPN (e.g., state space- and simulation-based approaches). In this article, we present a comparative analysis of test sequence generation approaches based on CPN models by means of literature reviews and a case study on medical systems: electrocardiography and insulin infusion pump. The results presented in this article provide insights for testers who need to select the most adequate test sequence generation approach when applying the MBT using CPN. The state space analysis and exploration guided by model checking showed to be the most cost-effective approach during our empirical evaluation.

Keywords: Model-based testing, abstract tests, formal specifications, Petri nets.

Lista de Figuras

Lista de Tabelas

Conteúdo

Capítulo 1

Introdução

Linguagens naturais e semi-formais são comumente aplicadas para documentar artefatos de software, ao custo de gerar possíveis ambiguidades durante a interpretação e verificação de requisitos. Por outro lado, linguagens formais de especificação (por exemplo, redes de Petri coloridas (CPN) [1]) são ferramentas matemáticas que se apóiam na teoria e na lógica de conjuntos para representar requisitos de software de maneira inequívoca, impedindo decisões de desenvolvimento enganosas. A especificação inequívoca é crítica para sistemas nos quais falhas podem levar a danos físicos, financeiros e ambientais, como sistemas médicos e aviônicos (conhecidos como sistemas críticos seguros). Dar suporte ao processo de desenvolvimento e certificação de sistemas dessa natureza é por si só um trabalho importante e necessário, dado que as propriedades de segurança devem ser imprescindíveis em sistemas com essa característica.

CPN é uma ferramenta matemática que pode ser utilizada para modelar graficamente a estrutura de um sistema. O termo redes de Petri denomina uma técnica de especificação formal adequada para modelagem de sistemas com atividades paralelas, concorrentes, assíncronas e não determinísticas [2]. É um modelo bastante utilizado principalmente em modelagem e validação de sistemas críticos seguros.

Modelos formais podem ser reutilizáveis durante várias atividades do processo de desenvolvimento. Por exemplo, os testadores reutilizam modelos para gerar testes abstratos considerando o teste baseado em modelo (*Model-Based Testing - MBT*). Neste trabalho, são seguidas as definições fundamentais de teste apresentadas por Li e Offutt [3]. Testes abstratos podem ser descritos como seqüências de transições relacionadas aos requisitos de teste

impostos por critérios de cobertura. Um exemplo de critério de cobertura para grafos é a cobertura de arestas, exigindo que os testadores incluam cada aresta do grafo no conjunto de requisitos de teste.

É possível observar que boa parte da atividade de teste é gasta na busca em identificar o que o sistema deveria fazer [4]. Ou seja, antes de se perguntar se o resultado está correto, deve-se saber qual seria o resultado correto. Um modelo é importante nessa tarefa, pois, se bem desenvolvido, captura o que é essencial no sistema. Há casos em que o modelo pode ser utilizado como um oráculo, definindo a linha que separa o comportamento adequado do errôneo. Os benefícios da modelagem formal para testes de software são notáveis, no entanto, é importante ponderar quando há a necessidade de utilizar todo o seu potencial, para não tornar o processo oneroso e desnecessário.

A atividade de teste de software é considerada por muitos especialistas uma etapa essencial no processo de desenvolvimento de sistemas. Porém, ainda há barreiras para sua utilização em projetos, como, por exemplo, os custos e o aumento em prazos de entrega. Com a intensão de auxiliar a aplicação da prática de testes e reduzir custos, a geração automática de testes é uma importante ferramenta de apoio. É possível encontrar soluções bem aceitas na literatura para a geração de casos de teste, na quais são identificados os cenários válidos para testes, entradas válidas e saídas esperadas. Na geração de dados de teste, o foco é fornecer as entradas necessárias para o teste ser executado. Na geração de sequências de teste, deseja-se garantir a cobertura dos requisitos do sistema e derivar casos de testes (combinação de dados e sequências) mais assertivos. Neste contexto, existem ferramentas disponíveis no mercado para gerar testes de maneira automatizada, como, por exemplo, a ferramenta Randoop[1] para geração de testes de unidade na linguagem Java. Existem benefícios em se utilizar esse tipo de ferramenta, porém, é possível também que testes desnecessários ou inúteis sejam produzidos.

Para preencher uma lacuna de pesquisa encontrada a partir de uma revisão terciária, neste trabalho, foram analisadas as abordagens existentes usadas para gerar testes abstratos usando modelos CPN de sistemas críticos seguros. A linguagem CPN é uma rede de Petri de alto nível usada para modelar sistemas compostos por processos paralelos, comunicação, sincronização, recursos compartilhados e processamento de dados [5]. CPN estende as redes de

[1] https://randoop.github.io/randoop/

Petri de baixo nível, incluindo tipos de dados, funções e módulos. Cenários de aplicações de CPN incluem, mas não se limitam a, protocolos de comunicação, sistemas operacionais, projetos de hardware, sistemas embutidos e processos de negócios.

Testadores geralmente conduzem a geração de testes abstratos com base em modelos CPN usando três abordagens gerais: (1) manipulando a estrutura do modelo (*e.g.*, representação XML); (2) explorando o espaço de estados do sistema em teste; e (3) realizando simulações de modelos CPN. Algoritmos e técnicas, como, por exemplo, busca em profundidade (*Depth-First Search - DFS*), o carteiro chinês e a verificação automática de modelo (*model checking*) suportam implementações específicas dessas abordagens gerais [7, 9, 10].

1.1 Motivação

Por meio de uma revisão terciária (revisão de revisões sistemáticas) realizada neste trabalho (Capítulo 3), foi identificado um número considerável de estudos com foco na análise do uso de linguagens e ferramentas de especificação para conduzir o MBT. Por exemplo, Bernardino *et al.* [11] fornece uma visão geral sobre ferramentas e modelos de MBT, realizando um mapeamento sistemático da literatura; enquanto Uzun e Tekinerdogan [12] conduzem uma Revisão Sistemática da Literatura (RSL) para analisar o MBT, identificando soluções propostas e direções atuais de pesquisa. Entretanto, ainda existe uma lacuna de pesquisa em relação à análise de abordagens baseadas em CPN para geração de testes abstratos.

1.2 Objetivos

Neste trabalho, a aplicação de abordagens existentes para geração de testes abstratos é discutida meio de uma análise empírica usando uma RSL e um estudo de caso sobre sistemas médicos de eletrocardiografia (ECG) e bomba de infusão de insulina. Por um lado, a RSL é realizada com base nas diretrizes de Kitchenham *et al.* [13] e Wohlin [14], e auxiliada pelo software Start© [2] para definir o protocolo de revisão e gerenciar dados. Por outro lado, o estudo de caso sobre sistemas de aquisição de ECG e de bomba de infusão de insulina foi realizado com base nos modelos de CPN descritos, verificados e validados em [6, 15].

[2] http://lapes.dc.ufscar.br/tools/start_tool

O objetivo geral com este trabalho é realizar uma discussão sobre a aplicação de técnicas existentes para a geração de testes abstratos para sistemas críticos seguros, fornecendo informações relevantes que possam auxiliar testadores em suas escolhas pela abordagem de geração de testes abstratos que é mais apropriada para um cenário de aplicação. Alguns objetivos específicos são importantes para a condução desse estudo:

1. realizar uma discussão sobre as técnicas mais utilizadas em modelos CPN para geração de testes abstratos;

2. realizar uma discussão sobre as técnicas para geração de testes abstratos, considerando modelos CPN e sistemas críticos seguros, por meio de um estudo de caso sobre sistemas médicos;

3. realizar uma análise de cobertura de requisitos dada a aplicação das técnicas para geração de testes abstratos, considerando os modelos CPN de um sistema de bomba de infusão de insulina e um sistema de aquisição de ECG.

1.3 Contribuições

É importante destacar as principais contribuições obtidas com este trabalho. A primeira delas, é uma revisão geral de revisões existentes (conhecida como revisão terciária) sobre MBT, na qual foi identificada a lacuna de pesquisa relacionada com CPN. Além disso, também é fornecida uma análise empírica sobre abordagens para geração de testes abstratos baseadas em CPN. A análise foi guiada por uma revisão sistemática da literatura com o tema de geração de testes abstratos utilizando exclusivamente o formalismo de CPN, e também é considerada uma contribuição. Por fim, um estudo de caso sobre geração de testes abstratos em sistemas médicos fornece um panorama para testadores e pesquisadores que precisam de informações sobre prós e contras de cada abordagem disponível na literatura.

1.4 Metodologia

Para atingir os objetivos da pesquisa, inicialmente uma revisão terciária da literatura (revisão de revisões) foi realizada, fornecendo uma análise dos estudos já existentes que uti-

lizam a técnica MBT para geração de testes abstratos, no intuito de realizar um levantamento das abordagens e modelos formais são utilizados para este fim. Neste ponto, foi identificada a ausência de uma análise sobre abordagens que utilizam CPN para geração de testes abstratos.

A partir disso, uma análise empírica foi conduzida, inicialmente através de uma revisão sistemática da literatura, que teve como objetivo identificar trabalhos e abordagens que utilizam CPN como formalismo para geração dos testes abstratos. Em seguida, os trabalhos foram classificados com base em critérios pré definidos, com o objetivo de escolher os melhores candidatos para replicação e análise dos resultados, desse modo fornecendo uma visão ampla das abordagens disponíveis na literatura para geração de testes abstratos com o formaismo de CPN.

O restante deste trabalho está organizado da seguinte maneira. No Capítulo 2 são descritos conceitos fundamentais sobre MBT, CPN e sistemas relacionados com o estudo de caso. No Capítulo 3 é descrita a revisão terciária sobre MBT. No Capítulo 4 é apresentada a análise empírica sobre abordagens para geração de testes abstratos com base em modelos CPN. No Capítulo 5 são descritas conclusões e trabalhos futuros.

Capítulo 2

Fundamentação Teórica

O objetivo com esse capítulo é abordar os conceitos fundamentais dos temas que são a base do estudo proposto.

2.1 Teste Baseado em Modelos

O teste de software é um processo de investigação de um produto de software para identificar possíveis incompatibilidades entre os requisitos atuais e esperados do sistema [56]. Uma das principais motivações em se utilizar testes de software é conferir a qualidade e o comportamento esperado de um sistema. Diz-se que um software está correto se cada entrada válida para o sistema produzir uma saída de acordo com as especificações. [12].

Já consolidada na indústria, a prática de testar softwares utiliza diversos critérios disponíveis na literatura. Dentre as técnicas mais recentes em pesquisas e aplicação, se destacam os testes de mutação [53] e a geração automática de testes [52]. Além disso, existem categorias de teste de software, exemplos dessas categorias são os Teste de caixa branca (*White box testing [WBT]*), e Teste Caixa Preta (*Black Box Testing [BBT]*). O WBT é geralmente considerado como testes baseados em código, e o BBT como testes baseados em modelos e/ou na especificação [55].

Para o contexto deste estudo, as atividades a serem desenvolvidas se enquadram na classificação de testes de caixa preta, onde o desenvolvimento baseado em modelos é útil e os benefícios da técnica de testes baseados em modelos (*model-based testing [MBT]*) são aproveitados.

MBT é uma técnica que utiliza modelos formais ou semi-formais para de alguma forma melhorar a performance e a confiabilidade de Sistemas Sob Teste (SUT). É um tema de pesquisa em crescimento e tem apresentado diversas contribuições, se tornando uma técnica consolidada na literatura [11], [46]. Na prática, o MBT utiliza um modelo (por exemplo, um diagrama de estado UML ou um modelo de CPN) que especifica parcialmente os comportamentos de um sistema. A partir deles, testes abstratos são gerados para cobrir requisitos de teste definidos por algum critério de cobertura previamente estabelecido [3].

A fase de testes é uma importante etapa na construção do software. A intenção é verificar e validar se o sistema faz o que deveria fazer. No entanto, o teste é a construção mais demorada e dispendiosa do desenvolvimento de software. Inclusive, gerar informações para testar software é conhecido como um processo difícil. Existem muitas razões para não realizar esse trabalho manualmente: é um trabalho demorado, a intervenção humana pode causar erros, a cobertura de teste pode não ser realizada completamente, entre outros fatores [55].

Modelos já são de fato, aplicados com sucesso para descrever o comportamento e os requisitos de um sistema em desenvolvimento. No entanto, muitos engenheiros de testes ainda produzem *scripts*, testes ou casos de testes de uma maneira informal com base principalmente em suas experiências. Os testes baseados em modelos são uma alternativa apropriada para derivar artefatos de teste, uma vez que a ideia da técnica se baseia na automatização da atividade de testar o software. Inclusive, a prática pode ser útil na redução de custos dos testes e do software no geral, dado que a fase de testes costuma representar 30 a 60% do esforço de desenvolvimento do software [56], MBT pode ser um importante aliado nessa questão [54].

A técnica costuma ser bem aproveitada em sistemas críticos seguros, que são definidos como sistemas nos quais o mau funcionamento do software pode resultar em morte, ferimentos ou danos ao meio ambiente. Nesses casos, o alto nível de confiabilidade no software é de extrema importância. A técnica MBT é defendida como uma excelente alternativa para este tipo de sistema, pois os testes baseados em modelos derivam artefatos importantes que podem identificar falhas potenciais. Em geral, sistemas que precisam adicionar altos níveis de confiabilidade escolhem utilizar a técnica a fim de prover essa característica nos testes do SUT.

Os domínios de aplicação da MBT são os mais variados, grande parte ligados à segurança

de sistemas críticos. O domínio automotivo detém a maior parte das aplicações. Porém a literatura também aponta aplicações da técnica no ramo de estradas de ferro, robótica, nuclear, aeroespacial, automação, aviação, medicina e consumo de energia [46].

Existem diversas maneiras de conduzir a aplicação de MBT em SUTs, a maneira mais comum é utilizando grafos, que representam os estados e caminhos entre estados de um modelo. Dessa maneira, os critérios de teste são definidos baseados nas estruturas do grafo. Um exemplo disso é a cobertura de arestas, que exige que cada aresta do grafo seja coberta por pelo menos um teste, ou seja, o conjunto de requisitos de teste deve contemplar cada aresta do grafo [3]. Outro critério de teste baseado em grafos é a cobertura de par de arestas, que exige que todos os pares de arestas sejam cobertos por pelo menos um conjunto de testes. Os pares de arestas são conjuntos de dois elementos pertencentes ao grafo que representam a ligação entre os nós. Além destes critérios, a cobertura de nós do grafo, que exige que todos os nós sejam contemplados por pelo menos um conjunto de testes também pode ser utilizada.

2.2 CPN e Sistemas Médicos

Para facilitar a compreensão, a definição formal de uma *CPN* pode ser descrita em três partes, a primeira delas é tratada como a definição de uma *CPN* não hierárquica (*CPN* sem módulos); a segunda parte define um módulo de *CPN*; e por fim, a terceira parte define uma *CPN* hierárquica. A definição das partes está disponível em [1] e é descrita como segue:

- *CPN* não hierárquica é uma tupla de nove elementos $CPN = (P, T, A, \Sigma, V, C, G, E, I)$ na qual:

    1. P é um conjunto finito de lugares. Em outras palavras, a quantidade de lugares de uma *CPN* é finito.
    2. T é um conjunto finito de transições tal que $P \cap T =$. Ou seja, não existe algum elemento em T igual a algum elemento em P.
    3. $A \subseteq P \times T \cup T \times P$ é um conjunto de arcos direcionados. Isso significa que um arco pode ligar apenas um lugar a uma transição ou vice-versa. Não é possível usar um arco para ligar um lugar a outro lugar ou uma transição a outra transição.

4. Σ é um conjunto não vazio de cores. Isto é, os lugares de *CPN* possui algum tipo de dado (ou cor), o qual pertence a Σ.

5. V é um conjunto finito de variáveis tipadas tal que $Tipo[v] \in \Sigma$ para todas as variáveis $v \in V$. Ou melhor, existe uma cor associada a cada variável tal que essa cor é um elemento de Σ.

6. $C : P \rightarrow \Sigma$ é uma função de conjunto de cor que associa uma cor a cada lugar. Dito de outro modo, cada lugar da *CPN* possui uma cor.

7. $G : T \rightarrow EXPR_v$ é uma função de guarda que associa uma guarda a cada transição t tal que $Tipo[G(t)] = Bool$. Isto é, a cor de guarda de cada transição é um *booleano*.

8. $E : A \rightarrow EXPR_v$ é uma função de expressão de arco que associa uma expressão de arco a cada arco a tal que $Tipo[E(a)] = C(p)_{\text{MS}}$[1], no qual p é o lugar conectado ao arco a. Em outros termos, a cor associada a a deve ser a mesma cor associada a p, que é o lugar com o qual a está conectado.

9. $I : P \rightarrow EXPR_\theta$ é uma função de inicialização que associa uma expressão de inicialização a cada lugar p tal que $Tipo[I(p)] = C(p)_{\text{MS}}$. Ou seja, a cor dos valores de inicialização de um lugar deve ser igual a cor daquele lugar.

- Módulo de *CPN* é uma tupla de quatro elementos $CPN_M = (CPN, T_{\text{sub}}, P_{\text{porta}}, PT)$, na qual:

1. $CPN = (P, T, A, \Sigma, V, C, G, E, I)$ é uma CPN não hierárquica.

2. $T_{\text{sub}} \subseteq T$ é um conjunto de transições de substituição. Isso significa que uma transição de substituição é também uma transição.

3. $P_{\text{porta}} \subseteq P$ é um conjunto de lugares porta. Dito de outro modo, um porta é também um lugar.

4. $PT : P_{\text{porta}} \rightarrow IN, OUT, I/O$ é um conjunto de tipo porta que associa tipos de porta a lugares. Em outras palavras, uma porta deve possuir um dos seguintes tipos: $IN, OUT, I/O$.

[1] MS está relacionado a "multiconjunto".

- CPN hierárquica é definida como uma tupla de quatro elementos $CPN_H = (S, SM, PS, FS)$, na qual:

  1. S é um conjunto finito de módulos. Cada módulo é um módulo de Rede de Petri Colorida $s = ((P^s, T^s, A^s, \Sigma^s, V^s, C^s, G^s, E^s, I^s), T^s_{\text{sub}}, P^s_{\text{porta}}, PT_s)$. É necessário que $(P^{s_i} \cup T^{s_i}) \cap (P^{s_j} \cup T^{s_j}) = \theta$ para todo $s_i, s_j \in S$ tal que $i \neq j$.

  2. $SM : T_{\text{sub}} \rightarrow S$ é uma função de sub-módulo que associa um sub-módulo a cada transição de substituição. É necessário que a hierarquia de módulo seja acíclica, ou seja, um sub-módulo não pode possuir transição de substituição associada a um sub-módulo superior na hierarquia.

  3. PS é uma função de relação $porta\text{-}socket$ que associa uma relação $porta\text{-}socket$ $PS(t) \subseteq P_{\text{sock}}(t) \times P_{\text{porta}}{}^{\text{SM(t)}}$ a cada transição de substituição t. É necessário que $PT(p) = PT(p')$, $C(p) = C(p')$ e $I(p)\langle\rangle$ para todo $(p, p') \in PS(t)$ e $t \in T_{\text{sub}}$. Em síntese, a cada transição de substituição existe uma associação de uma relação $porta\text{-}socket$, de modo que o tipo do $socket$ deve ser igual ao tipo da $porta$ e a cor do $socket$ deve ser igual a cor da $porta$.

  4. $FS \subseteq 2^P$ é uma família de conjuntos de fusão (*fusion sets*) não vazios tal que $C(p) = C(p')$ e $I(p)\langle\rangle = I(p')\langle\rangle$ para todo $p, p' \in fs$ e todo $fs \in FS$. Em outras palavras, é possível fundir dois lugares de modo que a marcação de ambos sejam sempre iguais. São semelhantes às variáveis globais conhecidas em muitas linguagens de programação.

Resumidamente, CPNs são grafos bi-partidos que possuem lugares e transições. Como regra geral, lugares só podem ser ligados à transições e vice versa. Os lugares representam condições e as transições podem representar eventos. Para habilitar as transições, tokens são associados aos lugares do modelo. Por exemplo, quando um evento (transição) acontece um token é subtraído do lugar de entrada e adicionado no lugar de saída. A quantidade de tokens subtraídos no disparo da transição é definida pelo peso do arco vinculado. A simulação do modelo também é chamada de *token game*.

A aplicação prática de CPN tipicamente se baseia em uma combinação de simulação automática, visualização, análise do espaço de estados e análise de desempenho. Essas atividades em conjunto resultam em uma validação do sistema em consideração, no sentido de

que é possível justificar a afirmação de que o sistema tem as propriedades desejadas, e um alto grau de confiança e compreensão do sistema foi obtido [5].

Uma considerável parcela dos sistemas médicos existentes são sistemas críticos seguros. Exemplos de sistemas médicos incluem ECG e bombas de infusão de insulina. Sistemas de controle da bomba de infusão de insulina são usados para auxiliar no tratamento do diabetes, simulando o comportamento do pâncreas. O software embutido controla o sistema, com o objetivo de evitar situações de perigo [16]. Nesta seção, os principais conceitos de sistemas de bomba de infusão de insulina, sistemas de ECG e CPN são descritos com base em uma amostra de modelos CPN descritos, verificados e validados no trabalho apresentado por Costa *et al.* [15] e Sobrinho *et al.*[6].

Considerando um relatório do consórcio internacional de jornalistas de investigação [17], na última década, a administração de alimentos e medicamentos (*Food and Drug Administration - FDA*) dos Estados Unidos identificou mais de 5.4 milhões de eventos adversos relacionados ao mau funcionamento de sistemas médicos, resultando em quase 83.000 mortes e 1.7 milhões de feridos. Na Tabela 2.1 é apresentada a quantidade de pacientes lesionados por mau funcionamento em dispositivos médicos. Das lesões relatadas, 155.387 referem-se a sistemas de bomba de infusão de insulina, sugerindo a importância de um processo de desenvolvimento rigoroso para ajudar a diminuir o número de sistemas com defeitos.

Tabela 2.1: Quantidade de pacientes lesionados por mau funcionamento em dispositivos médicos.

Quantidade de pacientes	**Dispositivos**
103.104	Prótese de quadril
94.826	**Bomba de insulina com sensor**
78.172	Estimulador espinhal
60.795	Tela cirúrgica
60.561	**Bomba de insulina implantada**
59.457	Desfibrilador

Uma amostra do modelo de CPN proposta por Costa *et al.* [15] é ilustrado na Figura 2.1, composta por módulos de hardware e software. Em geral, o hardware do sistema de bomba de infusão de insulina compreende cartucho, bateria, botões e agulha; enquanto o software

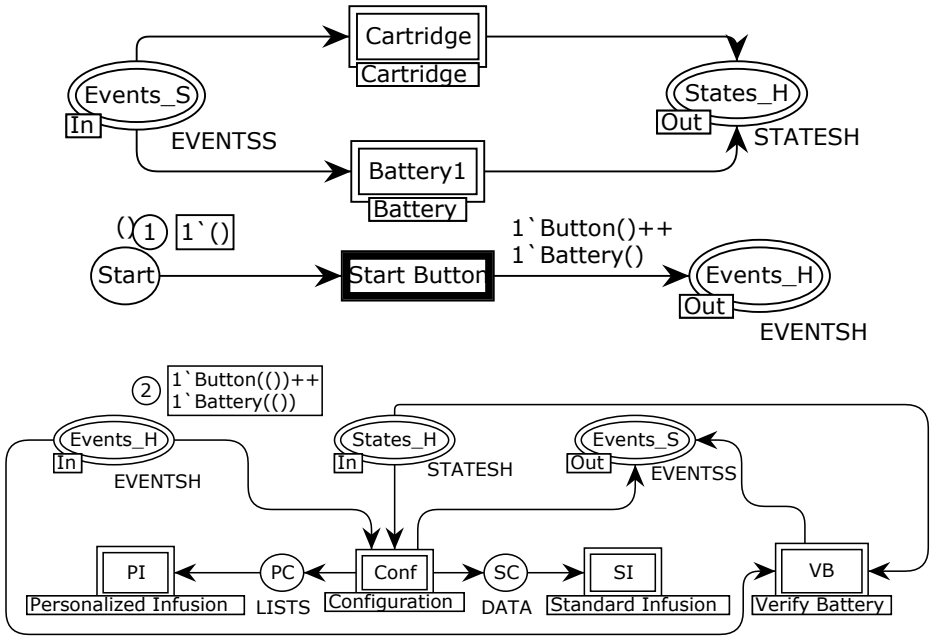


Figura 2.1: Amostra de simulação de módulos de hardware e software do modelo CPN de bomba de infusão de insulina apresentado em [15]

fornece recursos para configurar a bomba e controlar os comportamentos desejados, como a quantidade de insulina administrada. Isso também ilustra um dos principais recursos de hierarquia em CPN. A composição dos módulos (subpáginas ou sub-redes) permite a especificação de modelos hierárquicos, definindo transições de substituição (retângulos de borda dupla) e locais de entrada e saída (elipses de borda dupla).

Cada transição de substituição está relacionada a uma subpágina específica do modelo CPN. O módulo de software do sistema de bomba de infusão de insulina da Figura 2.1 contém subpáginas para representar a configuração da bomba, a verificação da bateria e os tipos de infusão (*i.e.*, infusão padrão e personalizada). Por exemplo, a subpágina *Standard Insulin* é ilustrada na Figura 2.2, contendo elementos como lugares (elipses), transições (retângulos), arcos (setas) e inscrições de rede. Modeladores somente podem conectar lugares a transições e transições a lugares, nunca lugares a lugares ou transições a transições. Os recursos básicos representados nesta subpágina são os tipos de infusão de insulina (*i.e.*, basal, bolus e bolus corretivo) e recarga de cartucho. Os estados dos sistemas são definidos como marcações do modelo CPN, representadas pela distribuição de fichas por todos os lugares no modelo. Na Figura 2.2, o lugar de entrada denominado *SCO* está associado a quatro fichas com o *colour*

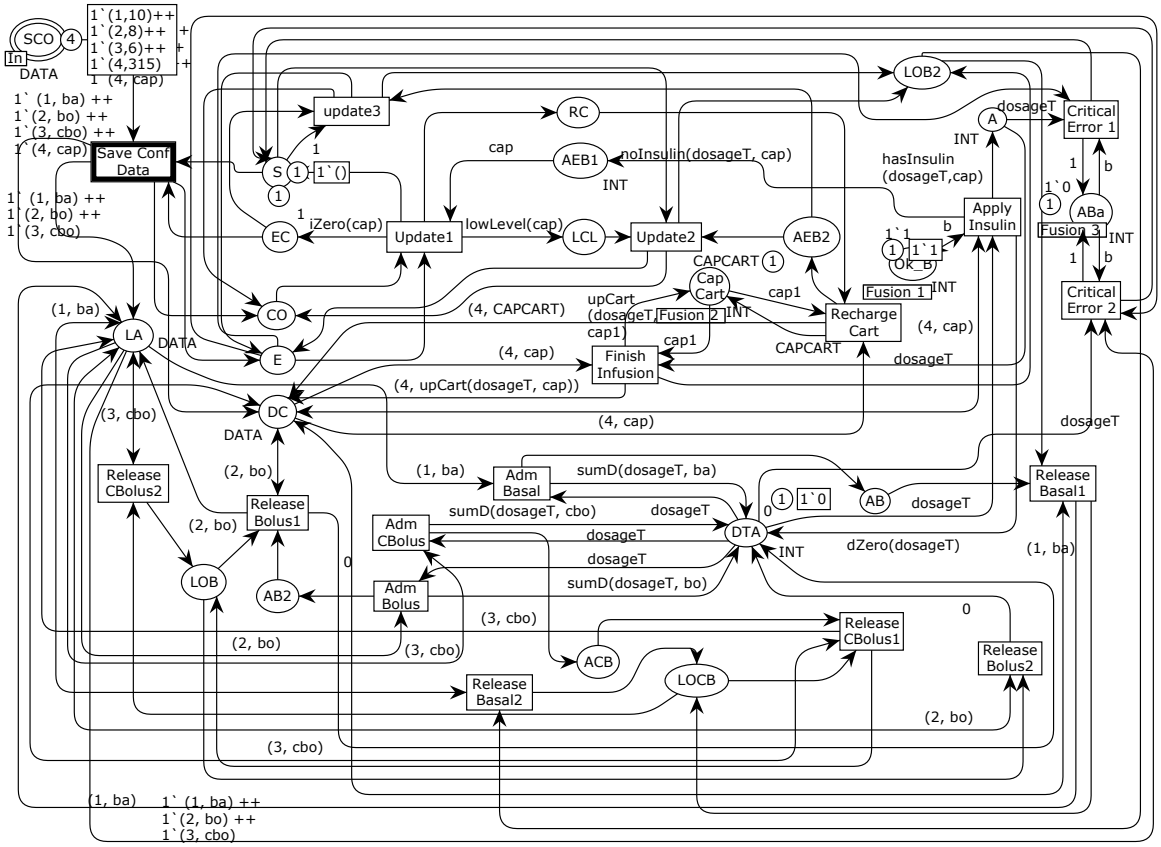


Figura 2.2: Subpágina *Standard Infusion* do modelo CPN para sistemas de bomba de infusão de insulina apresentado em [15]

set DATA (*i.e.*, duas tuplas de números inteiros).

O ECG é outro exemplo de sistema médico já modelado usando CPN [6]. O modelo é composto por dois módulos principais, que separam os estados e eventos relacionados a hardware e software. O modelo foi contruído de forma hierárquica, dessa forma, subpáginas vinculadas ao dois módulos principais podem sem acessadas nos níveis inferiores. Para o modelo completo do ECG, as subpáginas que controlam os eletrodos e bateria do equipamento estão vinculadas ao módulo de hardware, e a subpagina que representa a leitura dos dados e funcionamento do ECG é vinculado ao módulo de software. Na Figura 2.3 é apresentada uma amostra da simulação do modelo do módulo de software do modelo de ECG, e na Figura 2.4, as subpáginas que representam os eletrodos e bateria podem ser visualizadas. O módulo de bateria é responsável por realizar a recarga da bateria quando um evento de bateria é recebido do software. O módulo coleta os dados da bateria e envia um comando de recarga, caso seja necessário. De modo semelhante, o módulo de eletrodos é responsável por

receber e verificar a tensão dos eletrodos e coletar dados de impedância. Caso o sinal esteja dentro dos valores esperados, a leitura de sinal é liberada, caso contrário, uma nova leitura é realizada.

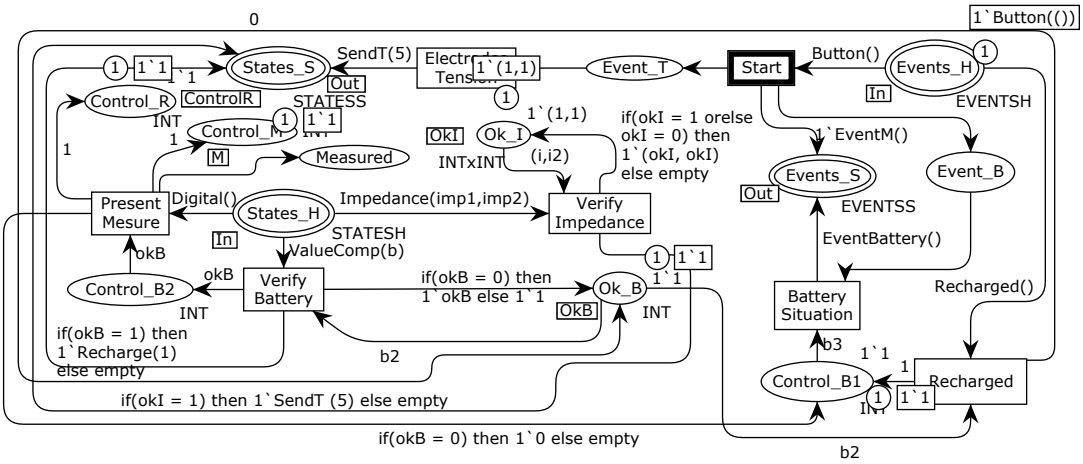
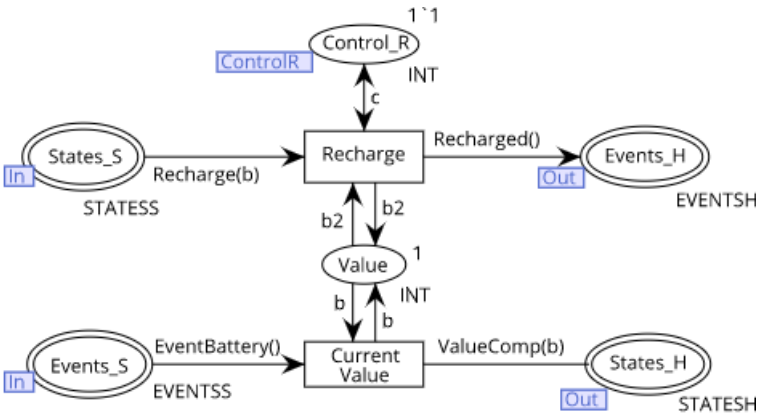
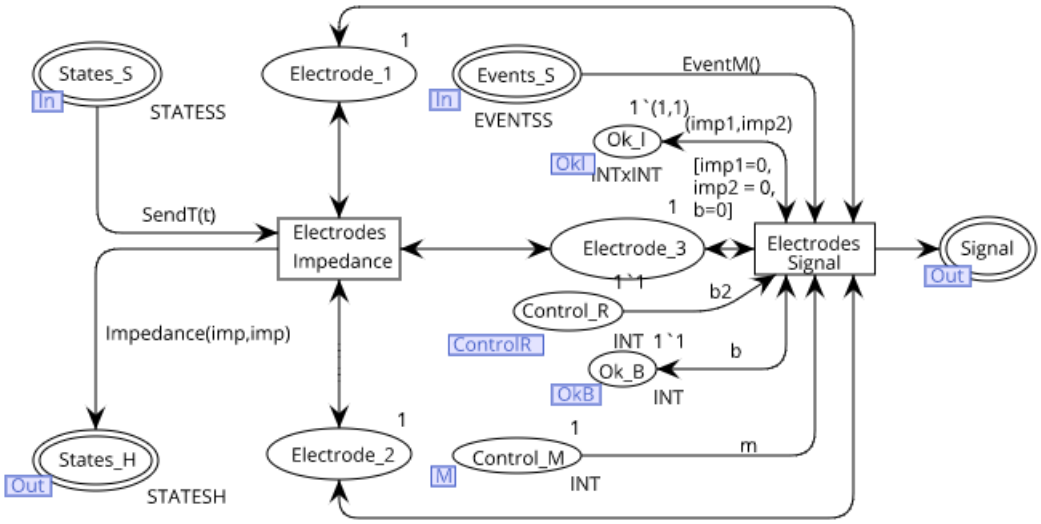


Figura 2.3: Amostra de simulação do módulo de software do modelo de ECG apresentado em [6].

Em geral, um sistema de ECG é composto por processadores, transdutores, amplificadores, filtros, conversores e software embutido (verificação dos níveis de impedância da bateria e dos eletrodos de pele). Em [6], os autores representaram componentes de hardware usando conceitos teóricos e algoritmos conhecidos, como a transformação rápida de Fourier, implementada usando a linguagem de programação CPN/ML. O CPN/ML é baseado no *Standard ML* (SML), uma linguagem modular e funcional para fins gerais. O livro de Jensen e Kristensen [5] pode ser consultado para leitores que precisam de descrições mais detalhadas dos conceitos de CPN.



(a)



(b)

Figura 2.4: (a) Amostra da subpágina que representa o funcionamento da bateria no modelo ECG apresentado em [6]. (b) Amostra da subpágina que representa o funcionamento dos eletrodos no modelo ECG apresentado em [6].

Capítulo 3

Revisão Terciária Sobre Teste Baseado em Modelos

A revisão terciária foi útil para descrever trabalhos relacionados, linguagens formais de especificação, modelos e lacunas de pesquisa relacionadas ao MBT. A revisão foi conduzida com base nas diretrizes de Kitchenham *et al.* [13].

3.1 Protocolo

Cinco etapas foram seguidas durante a revisão terciária usando o software Start©: (1) definição de questões de pesquisa, (2) definição de *strings* de busca e conjuntos de dados, (3) definição de critérios de inclusão e exclusão, (4) extração e classificação de estudos a partir dos conjuntos de dados e (5) análise e discussão de estudos selecionados. A Figura 3.1 ilustra as etapas seguidas nesta revisão, separando as etapas de planejamento e definição do protocolo, condução do estudo e resultados obtidos. No planejamento e definição do protocolo, as bases de dados, *strings* de busca, questões de pesquisa e critérios do estudo foram definidos. Na etapa de condução, as *strings* de busca foram aplicadas e os resultados analisados conforme os critérios estabelecidos. E por fim, na etapa de relatórios e resultados, ocorreu a classificação e análise dos dados para responder as questões de pesquisa.

Três questões de pesquisa (QP) orientaram a revisão: **(QP1)** quais são as linguagens de especificação formal mais usadas para o MBT? **(QP2)** quais são as ferramentas mais usadas para MBT usando as linguagens formais de especificação identificadas? e **(QP3)** quais são

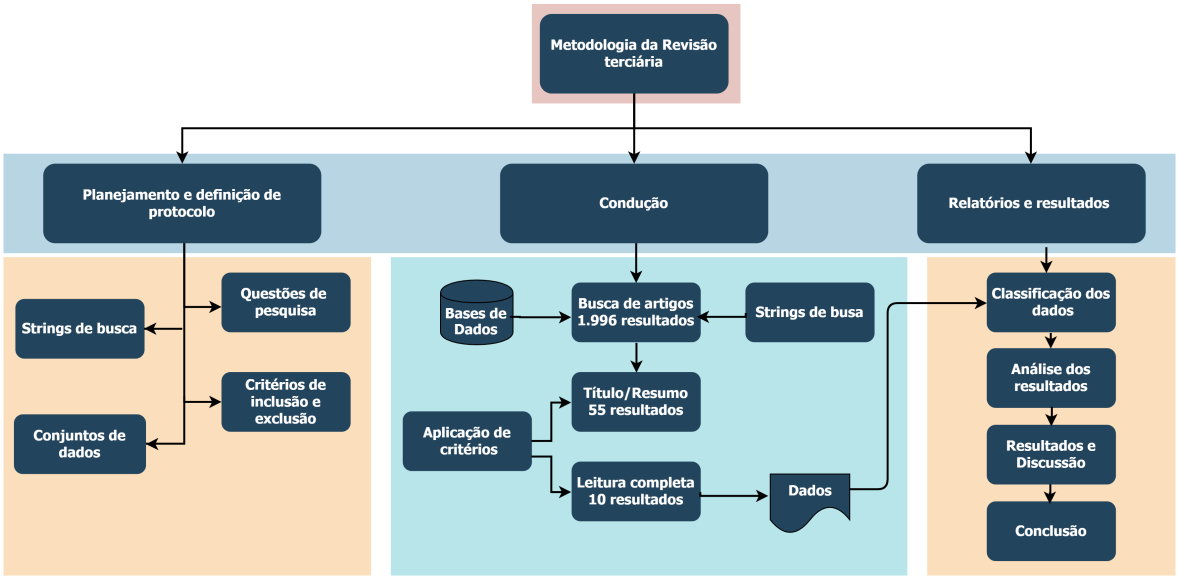


Figura 3.1: Etapas da revisão terciária.

as análises existentes focadas no MBT e no CPN?

Foram utilizadas as bases de dados e definidas as *strings* de busca apresentadas na Tabela 3.1. Os estudos consultados nas bases de dados foram importados para o software Start©. Além disso, os seguintes critérios de inclusão orientaram a revisão: (I1) o estudo está focado no MBT usando linguagens de modelagem formal, (I2) o estudo é uma revisão, (I3) o estudo foi publicado até junho de 2020 e (I4) o estudo está disponível para download; enquanto os seguintes critérios de exclusão orientaram a revisão: (E1) *posters* e resumos, (E2) estudos duplicados.

Tabela 3.1: Bases de dados e *strings* de busca para a revisão terciária.

Bases de dados	***Strings* de busca**
IEEE Xplore e *Web of science*	(((model based testing) OR model-driven testing) AND((literature OR systematic) AND (review OR map*)))
Science Direct, *ACM digital library* e *Springer*	(((formal(models OR methods OR specifications)) AND("model-based testing"OR "model-driven testing")) AND((literature OR systematic) AND (review OR map*)))

3.2 Resultados

A execução das *strings* de busca resultou em 1.996 artigos: 588 para IEEE Xplore, 635 para ACM Digital Library, 10 para Web of Science, 354 para Science Direct e 409 para Springer. Não foram aplicados filtros de data, com o objetivo de obter o número máximo de publicações sobre o tópico MBT a partir dos resultados da pesquisa. Após análise dos títulos e resumos, 55 artigos foram selecionados para a leitura dos textos completos. Por fim, foram aceitos 10 artigos para extração de dados.

A análise dos trabalhos resultou na identificação de linguagens formais de especificação utilizadas para o MBT, considerando as contribuições e limitações. A maioria dos trabalhos está relacionada a tendências no uso de modelos para especificar sistemas críticos (por exemplo, sistemas em tempo real) que exigem altos níveis de precisão.

Sabbaghi et al [48] realizam uma revisão sistemática sobre MBT aplicada a sistemas baseados em estados. Os pesquisadores destacam que muitos dos sistemas de software atuais são baseados em estado. O MBT é destacado como uma maneira eficiente de gerar testes, com a capacidade de revelar defeitos. Além disso, realizam um levantamento dos principais modelos que podem ser usados para auxiliar a geração automática de testes. Para esse tipo de cenário, os modelos classificados como mais adequados durante o estudo foram: máquinas de estados finitos (*Finite State Machines* (FSM)), máquinas de estados finitos estendida (*Extended Finite State Machines* (EFSM)), diagramas da máquina de estados UML, autômatos temporizados e cadeia de Markov.

Paiva e Simão [51] apresentam uma pesquisa focada na geração de testes com MBT utilizando o formalismo de Sistemas de Transição de Entrada e Saída (*Input/Output Transition Systems* (IOTS)). Os pesquisadores destacam os formalismos FSM e UML *Statecharts* como outras opções que podem ser usadas para modelar rigorosamente uma aplicação. Entretanto, defendem que IOTSs são modelos mais interessantes para a comunidade de pesquisa, uma vez que são mais expressivos, especialmente quando se trata de não-determinismo. Alguns dos benefícios no uso de IOTSs é que o formalismo não impõe restrições aos pares de entrada/saída como nas FSMs, e uma IOTS pode atingir um estado no qual nenhuma ação de saída é produzida. O formalismo também aceita entradas em qualquer estado, porém pode não ser adequado para modelar alguns tipos de sistemas (*e.g.*, sistemas interativos), uma vez

que os IOTS rejeitam entradas em algumas situações. No entanto, ainda podem ser usados para modelar comportamentos de processos, como especificações, implementações e testes. Além disso, IOTS é um formalismo adequado para modelar sistemas complexos, como protocolos de comunicação e sistemas distribuídos. Segundo os pesquisadores, para aplicar o MBT com o formalismo IOTS, é necessária a utilização de uma ferramenta para automatizar a geração dos testes. Algumas ferramentas disponíveis apontadas no estudo foram: *Test Generation with Verification* (TGV), TorX, UPAAAL e *Labelled Transition Systems* (LTS)-BT. Durante a pesquisa foram encontradas algumas classes variantes do formalismo com características levemente diferentes: IOLTS (um IOTS que não considera o recurso ativado por entrada), IOSTS (sistemas de transição simbólica), TIOTS (IOTS cronometrado) e MIOTS (Multi IOTS).

Bernardino *et al.* [11] apresentam um mapeamento sistemático da literatura sobre modelos e ferramentas utilizados em MBT. São descritos sete passos para a aplicação do MBT:

1. Escolha da linguagem ou notação a ser utilizada de acordo com a aplicação;
2. gerar as entradas de testes (scripts, casos de testes, dados de entrada da aplicação);
3. elaboração de um oráculo de testes;
4. executar os scripts de testes para cada caso de testes;
5. comparar os resultados dos testes com as saídas esperadas;
6. decidir ações futuras baseadas nos resultados;
7. Parar os testes.

Entre os principais questionamentos na pesquisa está a aplicação das ferramentas disponíveis, onde os autores classificaram os estudos selecionados como aplicação comercial, acadêmica e open-source. A maior parte das ferramentas evidenciadas nos estudos são de aplicação acadêmica (40 de 70 ferramentas). Também foi possível notar que grande parte das ferramentas utilizam FSM ou UML para modelagem, ou ainda uma combinação das duas notações, e que todas as ferramentas focam em testes funcionais.

Shafique *et al.* [47] descrevem uma revisão também abordando as ferramentas disponíveis para MBT. No entanto, devido às muitas linguagens de modelagem existentes, o foco foi

nas ferramentas em que algum tipo de linguagem de modelagem baseada em estados é usada. Os principais modelos considerados no protocolo de revisão foram: FSMs, EFSMs, máquinas de estados abstratos (*Abstract State Machines* (ASM)), *Statecharts*, autômatos (temporizados, entrada/saída), redes de Petri, diagrama de fluxo de estado e cadeias de Markov. Os autores encontraram 46 ferramentas MBT e 2 APIs. Doze ferramentas foram selecionadas para estudo primário baseados nos critérios do protocolo da revisão. Das ferramentas selecionadas, 8 delas são direcionadas a modelos de FSM/EFSM enquanto as 4 restantes se dedicam a UML. As ferramentas *Graph Walker*, *Spec explorer* e *CertifyIT* também foram reportadas.

Gurbuz *et al.* [46] apresentam uma discurssão sobre a importância dos testes de software em sistemas críticos seguros. A técnica MBT é defendida como uma excelente alternativa para este tipo de sistema. O principal objetivo é fornecer uma revisão para analisar e descrever os avanços com MBT no campo de segurança de software, especialmente na aplicação de testes de propriedades de segurança. Os artigos selecionados destacaram que os domínios de aplicação de MBT são os mais variados. O domínio automotivo detém a maior parte das aplicações. Porém, existem também aplicações de MBT em estradas de ferro, robótica, nuclear, aeroespacial, automação, aviação, medicina e consumo de energia. Também é possível verificar que grande parte dos métodos de pesquisa dos trabalhos são estudos de caso, embora haja alguns trabalhos com experimentos. Com relação ao tipo de evidência, os estudos acadêmicos possuem maior número frente aos estudos aplicados industrialmente. É importante destacar que, no estudo, são apresentadas motivações em diversos trabalhos que suportam a utilização de MBT, como, por exemplo, a redução de tempo e custo de desenvolvimento, melhoria da cobertura de testes, melhoria da eficiência e da qualidade dos testes e o aumento da detecção de falhas. Outra questão importante respondida é que 42 porcento dos estudos apontaram que a notação mais utilizada pela técnica são autômatos e suas variações. Em segundo lugar fica a linguagem específica de domínio (*Domain Specific Language* (DSL)), seguida da UML. Com relação à utilização, a maior parte dos trabalhos foca em gerar casos de teste, seguidos de sequencias de teste e dados de teste. Os desafios encontrados pelos autores sugerem que as direções das pesquisas apontam para a possibilidade de fornecer uma abordagem da técnica de propósito geral, sem considerar um domínio específico de aplicação. Também há dificuldades para lidar com a representação de mudanças de contexto. O

comportamento de alguns sistemas críticos são particularmente sensíveis ao contexto, o que pode gerar uma especificação complexa que dificulte a aplicação de MBT.

Uzun e Tekinerdogan [12] focam em uma tendência recente no MBT, que adota modelos de arquitetura de software para fornecer suporte automatizado para o processo de teste. Isto é realizado levando em consideração à noção de teste baseado em arquitetura orientada a modelos (*Model-Driven Architecture Based Testing* (MDABT)). A tendência é definida pelos autores como uma técnica que adota modelos de arquitetura de um sistema em teste e/ou seu ambiente para derivar artefatos de teste. No artigo, a técnica original também é abordada. Os autores relatam os principais benefícios encontrados na literatura: melhor cobertura de teste, tempo de teste reduzido, maior confiabilidade, reutilização de testes, maior confiança no sistema, menos esforço humano, redução de custos, maior detecção de falhas e melhor qualidade do produto. Exemplos de modelos mais utilizados para esse fim incluem FSMs, redes de Petri, autômatos de E/S e cadeias de Markov. É descrito que o critério de teste mais utilizado no MBT é a cobertura de código. Nessa revisão, para a geração de casos de testes, LTS, Redes de Petri e UML *Testing Profile* (UTP) parecem ser os modelos preferidos na literatura. A maioria dos estudos usa modelos de teste baseados em grafos ou autômatos, em que casos de teste são gerados a partir de caminhos extraídos entre nós. Entre as direções das pesquisas sobre o tema, destaca-se a preocupação com a explosão de estados, necessidade de aplicar a MDABT em sistemas reais complexos, já que grande parte dos estudos são exemplos de menor complexidade ou estudos de caso. além disso, destacam a necessidade de automatizar o processo e sua análise de custo-benefício.

Saeed *et al.* [49] apresentam uma revisão sistemática sobre aplicações experimentais de técnicas fundamentadas em pesquisa para testes baseados em modelos . Quatro taxonomias são propostas para classificar as aplicações, além de classificar os estudos em uma série de categorias, como, por exemplo, tipos de teste, domínios de aplicação, nível dos testes e proposta de testes. Os resultados encontrados informam que grande parte dos trabalhos experimentais na área se concentram na categoria de testes funcionais e estruturais. Porém, é possível encontrar a aplicação da técnica em trabalhos de testes de regressão, testes de *stress* e testes de interface. O domínio das aplicações é em sua maior parte, em sistemas embarcados. É importante destacar que outras áreas também aparecem no estudo, como aplicações baseadas em redes, aplicações web, sistemas de tempo real e linhas de produtos de software.

Com relação aos modelos, máquinas de estado e modelos de simulação aparecem com maior parte dos estudos. FSMs e EFSMs aparecem também com quantidade significativa. Além disso, alguns trabalhos reportados utilizam diagramas UML. Como direções futuras de pesquisa, é levantada também a importância da aplicação industrial, uma vez que a maior parte dos trabalhos na área ainda são de caráter experimental. Utilizar técnicas híbridas surgem também como uma alternativa para aplicação em determinados tipos de sistemas.

Siavashi *et al.* [50] realizam uma revisão sistemática sobre a utilização de modelos de ambiente em MBT e seus benefícios para a técnica. Para os pesquisadores, a abstração é benéfica em esconder detalhes desnecessários da implementação e reduzir a complexidade do teste. No entanto, é também essencial que um modelo de teste seja detalhado o suficiente para gerar casos de teste efetivos. Os autores ainda destacam que encontrar o nível certo de abstração para o modelo de teste é um dos desafios em MBT. Além de discutir as características encontradas nos estudos primários dos modelos de ambiente em MBT, destacam as vantagens em utilizar os modelos na técnica, como, por exemplo a criação de oráculos de testes, geração automática de testes, otimização, redução do tamanho de espaço de estados e validação de requisitos com antecedência. Com relação aos formalismos mais utilizados para a combinação das técnicas, são destacados UML, autômatos temporizados, gramática de eventos atribuídos (*Attributed Event Grammar* (AEG)) e redes de Petri.

Petry et. al [44] conduzem o trabalho em torno do tema MBT para linhas de produtos de software. Os autores oferecem um panorama do estado da arte, realizando um mapeamento sistemático da literatura para responder perguntas relacionadas a domínios, abordagens, tipos de solução, variabilidade, automação de caso de teste, artefatos e avaliação. Os principais resultados obtidos pelos pesquisadores consideram os domínios de software e automotivo os que mais aplicam a técnica. O teste de caixa preta é amplamente realizado e a maioria dos estudos tem suporte totalmente automatizado. Para o contexto de linhas de produtos de software, máquinas de estados finitos é o modelo mais usado. Estudos de caso e experiências são usados para avaliar soluções de MBT e a maioria é realizada em ambientes industriais. Um ponto importante identificado é que a rastreabilidade não é amplamente explorada para soluções MBT.

Neto *et al.* [41] realizaram um levantamento sobre abordagens, tipos de evidência, nível de teste e domínios de aplicação. Dentre os estudos, alguns apontam EFSM, FSM e ASM

como modelos formais utilizados, mas a grande maioria utiliza diagramas UML. Os autores identificaram que a maior parte dos estudos se dedica ao nível de teste de sistema. Mas é possível encontrar aplicações do MBT em testes de integração, testes de unidade e também de regressão, embora a aplicação nesses níveis sejam menos frequentes.

Com base nos estudos primários selecionados para essa revisão, algumas perguntas de pesquisa precisam ser respondidas:

Respondendo a QP1

A maior parte dos trabalhos selecionados pela revisão evidenciam uma forte tendência para a utilização de FSM e EFSM , embora máquinas de estados abstratos (ASM) , Máquinas de estado da UML, Autômatos (temporizados, entrada/saída), Harel statecharts , Redes de Petri , Diagrama de fluxo de estado DSL, Gramática de Eventos Atribuídos (AEG), e cadeias de Markov também apareçam como modelos válidos para esse fim. Diversas outras notações também aparecem nos estudos, porém com menos frequência.

O gráfico na Figura 3.2 é utilizado para ilustrar as notações mais utilizadas levantadas pela revisão:

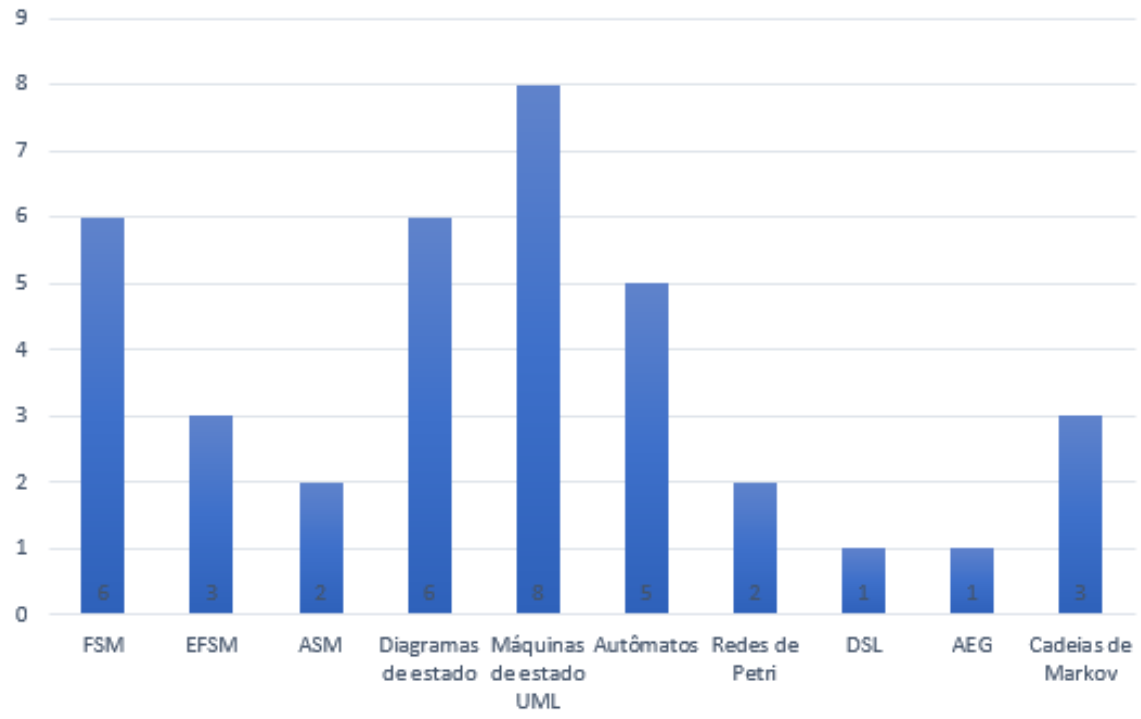


Figura 3.2: Principais notações identificadas nos estudos.

Respondendo a QP2

Os estudos sugerem que há diversas ferramentas disponíveis no meio acadêmico para aplicação da técnica MBT e que a escolha delas está diretamente ligada com a notação escolhida. Em [11], onde o trabalho consiste em identificar os modelos e ferramentas utilizados pela técnica, dentre as ferramentas encontradas, algumas estão listadas abaixo seguidas das notações suportadas:

1. Spec Explorer (ASM, Spec, FSM);
2. Qtronic (QML, UML, SMs com blocos de Java ou C#);
3. Test Generation with Verification Technology (IOLTS);
4. Alloy Analyzer (Notação Z/Alloy);
5. Graph Walker (EFSM e FSM);
6. Model-based integration and system test automation (Redes de Petri).

Também há uma classificação entre ferramentas que utilizam UML, FSM e uma terceira categoria denominada OT (others) onde foram concentradas todas as ferramentas encontradas que suportam notações diferentes. Entretanto, em [51] as ferramentas TGV, TorX, UPAAAL e LTS-BT também aparecem entre as mais utilizadas. Outras ferramentas que são evidenciadas no estudo [50] são UML Tools, Promela (SPIN), Markov Models(Cadeias de Markov) e TINA (Redes de Petri).

A maioria dos autores dos trabalhos selecionados declara o uso de FSM e EFSM, embora o ASM, autômatos temporizados, redes de Petri convencionais e cadeias de Markov também apareçam como linguagens formais de especificação geralmente aplicadas ao MBT. Várias ferramentas foram abordadas nos trabalhos selecionados. À medida que as pesquisas avançavam ao longo dos anos, foi possível verificar o crescimento do uso de novas ferramentas e linguagens de especificação para o MBT; no entanto, ainda não abrange algumas notações (por exemplo, CPN). Isso demonstra que o tópico MBT permanece aberto para novas pesquisas em diferentes linguagens formais de especificação.

Os dados utilizados para responder à essa questão de pesquisa foram extraídos principalmente dos trabalhos [11] e [47], que focam em ferramentas e formalismos para aplicação de

MBT, uma vez que nem todos os trabalhos selecionados tem este assunto como objetivo de pesquisa.

Respondendo a QP3

Nenhuma das revisões de literatura identificadas abordou a condução do MBT usando a linguagem de especificação CPN. Portanto, uma lacuna de pesquisa relevante encontrada durante a revisão das revisões sobre MBT é destacada: *a falta de uma análise das abordagens abstratas de geração de testes para conduzir o MBT usando modelos projetados com CPN.*

Capítulo 4

Análise Empírica

Neste capítulo, uma Revisão Sistemática da Literatura (RSL) sobre MBT com CPN foi utilizada para realizar uma comparação de abordagens para geraração de testes abstratos. Para isso, a realização de um estudo de caso sobre o modelo de sistema de bomba de infusão de insulina descrito por Costa *et al.* [15] e o modelo de ECG descrito por Sobrinho et al [6] foram utilizados para aplicar as abordagens propostas em estudos selecionados durante a RSL.

4.1 Revisão Sistemática da Literatura

Considerando a lacuna de pesquisa sobre MBT com CPN identificada no Capítulo 3, neste trabalho, o software Start© também foi usado para conduzir uma RSL com base nas diretrizes de Kitchenham *et al.* [13] e Wohlin [14].

4.1.1 Protocolo

Também foram realizadas cinco etapas durante a RSL: (1) definição de questões de pesquisa, (2) definição de *string* de pesquisa e bases de dados, (3) definição de critérios de inclusão e exclusão, (4) extração e classificação dos estudos a partir das bases de dados e (5) análise e discussão dos estudos selecionados.

Três questões de pesquisa orientaram a revisão: **(QP4)** quais são as abordagens usadas para gerar testes abstratos com base em CPN? **(QP5)** quais são as vantagens e desvantagens

das abordagens identificadas? e **(QP6)** como as abordagens influenciam a quantidade e a qualidade dos testes abstratos para sistemas críticos seguros?

A *string* de pesquisa *("coloured Petri nets"OR "colored Petri nets"OR "CPN") AND ("test sequences"OR "test cases"OR "abstract tests")* foi utilizada para executar as pesquisas iniciais usando as bibliotecas IEEE Xplore, Science Direct, Web of Science, Springer e ACM Digital Library. Posteriormente, pesquisas manuais usando o Google Scholar foram realizadas e a técnica *backward snowballing* foi aplicada nos estudos para melhorar a confiança na RSL.

Os critérios de inclusão e exclusão também orientaram a revisão. Os critérios de inclusão incluíram (**I1**) pesquisas com foco em gerar testes abstratos usando CPN e (**I2**) pesquisas publicadas até junho de 2020. Os critérios de exclusão incluíram (**E1**) artigos e pôsteres resumidos (**E2**) pesquisas duplicadas, (**E3**) pesquisas que não estão disponíveis para download, (**E4**) pesquisas que dependem de técnicas ou ferramentas externas, (**E5**) pesquisas que usam outras extensões de redes de Petri, (**E6**) pesquisas que apresentam revisões sistemáticas e (**E7**) pesquisas que não apontam explicitamente a técnica usada para gerar testes abstratos. As pesquisas também foram classificadas para poder selecionar estudos nos quais as abordagens de geração de testes abstratos são o mais fácil possível de replicar durante o estudo de caso apresentado na Seção 4.2. Três atributos possibilitaram a classificação das pesquisas: (1) nível de clareza na apresentação da abordagem, (2) relação do cenário de aplicação com sistemas críticos seguros e (3) integridade da descrição dos algoritmos.

O protocolo RSL compreendia cinco campos de extração de dados para orientar a resposta ás perguntas de pesquisa. Mais especificamente, esses campos de extração incluíram data de publicação, origem do artigo, cenário de aplicação, algoritmo e abordagem abstrata de geração de teste.

4.1.2 Resultados

Na Figura 4.1 é ilustrada uma visão geral do processo de busca e dos resultados da RSL. Durante a primeira busca, os conjuntos de dados retornaram 2.683 estudos: *IEEE Xplore Library* (19), *Science Direct* (174), *Web of Science* (28), *Springer* (349), *ACM Digital Library* (43) e Google Acadêmico (2070). Inicialmente, foram lidos os títulos dos 2.683 trabalhos obtidos, seguidos dos resumos, quando necessário. O primeiro filtro resultou em 47 pes-

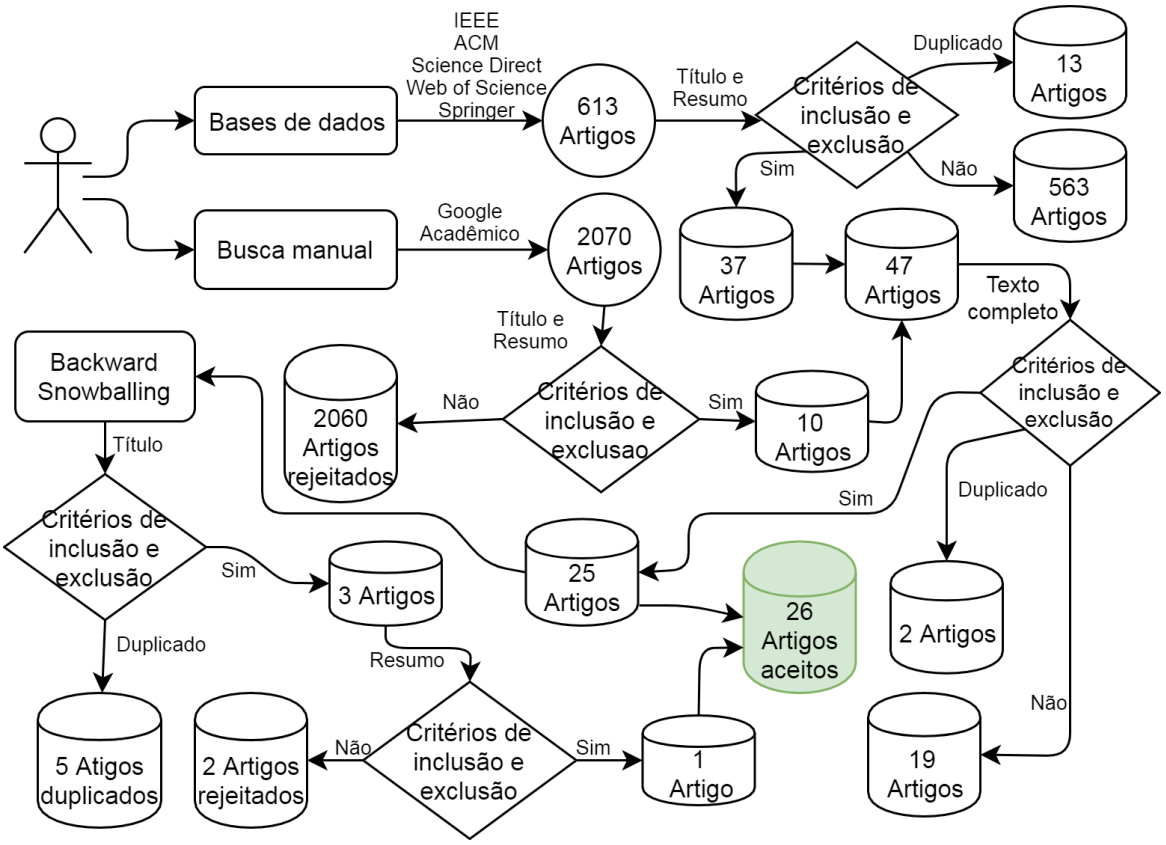


Figura 4.1: Visão geral do processo de busca e resumo dos resultados.

quisas selecionadas. No segundo filtro, os trabalhos foram lidos completamente, resultando em 25 estudos. Além disso, a técnica *backward snowballing* foi aplicada para aumentar a confiança na completude das buscas, encontrando mais um estudo. Foram considerados os critérios de inclusão e exclusão ao aplicar todos os filtros para selecionar os 26 artigos finais.

Ao ler as pesquisas, foram identificadas as principais abordagens para gerar testes abstratos usando CPN. Foram considerados pontos positivos e negativos de cada abordagem durante a leitura dos artigos. A maioria das pesquisas concentra-se em sistemas críticos seguros como cenários de aplicação.

Sun *et al.* [18] concentram-se em sistemas paralelos, usando a abordagem da análise do espaço de estados para gerar testes abstratos. Os autores também aplicam técnicas para ajudar a prevenir o problema de explosão do espaço de estado. Os resultados não continham testes abstratos inúteis ou redundantes. No entanto, a abordagem é aplicável apenas a sequências de comportamento linear; se não for linear, uma conversão é necessária. Ainda no contexto de sistemas paralelos, Sun *et al.* [19] propõem um método para gerar testes abs-

tratos com base em CPN, visando a cobertura total de ocorrências em diferentes ordens para todos os comportamentos em teste, sem testes redundantes usando a abordagem de espaço de estados. Os autores aplicam uma técnica de remoção de estado para reduzir o espaço de estado e a redundância. Algumas definições precisam ser obedecidas e uma metodologia de modelagem específica é apresentada, podendo desestimular seu uso. Sun *et al.* [20] apresentam um estudo de caso usado para descrever uma abordagem para reduzir modelos para gerar testes abstratos para sistemas paralelos. A abordagem reduz o tamanho do modelo CPN original, preservando os comportamentos originais. Como o número de estados em sistemas paralelos geralmente é grande, a abordagem de espaço de estados pode cobrir os objetivos do teste usando menos testes abstratos. A abordagem sugere que os modelos precisam ter comportamento linear para a técnica funcionar corretamente. Sun *et al.* [21] propõem um algoritmo para gerar testes abstratos com base na abordagem do espaço de estados. Os autores consideram um exemplo de sistema de aprovação de contratos para validar a abordagem, mostrando que nenhuma redundância é incluída nos testes abstratos gerados. Porém, em virtude do estudo de caso escolhido é difícil observar como a abordagem se comporta em modelos com espaço de estados maiores.

Zhao *et al.* [9] aplicam o algoritmo do carteiro chinês e a abordagem de simulação do modelo CPN para gerar testes abstratos. O espaço de estado e a análise de arquivo XML são usados para verificar o modelo e gerar casos de teste, respectivamente. Os autores afirmam que a abordagem previne testes abstratos redundantes, laços infinitos e o problema de explosão de espaço de estado. Como cenário de aplicação, os autores especificam um modelo CPN de um sistema de controle ferroviário e geram casos de teste a partir do modelo, que precisa obedecer algumas regras de modelagem para contemplar os pré requisitos necessários para a abordagem.

Cheng *et al.* [22] conduzem um estudo considerando um subsistema de bordo derivado de um sistema de controle ferroviário, comparando um algoritmo proposto com o algoritmo tradicional de busca em profundidade (*Depth First Search (DFS)*). O objetivo principal é gerar um conjunto de casos de teste com o menor número possível de caminhos repetidos, usando a abordagem de espaço de estado. Os autores argumentam que a abordagem trata de laços complexos e evita o problema de explosão do espaço de estado. Entretanto, uma metodologia de modelagem é necessária, e regras como nomenclatura de variáveis, lugares

e colorsets são exigidos para que o algoritmo se comporte da forma esperada. Zhang *et al.* [10] também realizam um estudo considerando um sistema de controle ferroviário. A verificação automática de modelos é usada para gerar um conjunto de testes abstratos para validar se um sistema está em conformidade com as propriedades de segurança. Todos os estados alcançáveis são analisados para detectar violações de propriedades. Os autores afirmam que a abordagem pode reduzir tempo de desenvolvimento, garantindo a conformidade entre o sistema e as propriedades. O modelo utilizado no estudo de caso, apesar de ser considerado crítico, é relativamente simples, tornando os resultados em modelos mais complexos pouco previsíveis. Zheng *et al.* [7] aplicam os algoritmos *paths covered optimal Sequence priority selected - SPS* para gerar testes abstratos para um sistema de blocos, como parte de um sistema de controle ferroviário. O algoritmos são aplicados no arquivo XML que representa o modelo CPN. Os autores comparam os algoritmos propostos com o algoritmo RW-TSG e DFS tradicional, mostrando que os algoritmos propostos obtém melhores resultados. A abordagem não gerou seqüências inúteis ou redundantes durante a avaliação, mas para garantir esses resultados, uma série de conjuntos de dados são necessárias, provenientes de uma metodologia específica de modelagem.

Zheng e Hu [8] usaram a mesma abordagem do trabalho [7], também para um sistema de controle ferroviário. A abordagem combinada de espaço de estados/estrutural apresentada conduz um método automatizado de otimização de sequências de teste. Os autores integram o algoritmo de labirinto ao algoritmo de colônia de formigas para produzir as sequências ideais. Com as melhorias nos algoritmos, os menores caminhos de busca são encontrados e as seqüências de teste redundantes são reduzidas, baseadas na natureza do algoritmo da colônia de formigas. Os autores afirmam que comparando os resultados com o algoritmo RW-TSG e com os algoritmos DFS tradicionais, o algoritmo proposto produz sequências e testes mais eficientes, no entanto, essa comparação não é disponibilizada no artigo.

Lijie *et al.* [23] aplicam a abordagem de espaço de estado, gerando testes abstratos com base no caminho para o estado de destino usando modelos CPN. A abordagem proposta foca no sistema de controle ferroviário chinês (Nível 3). Os autores avaliam a abordagem considerando as especificações técnicas do sistema de controle ferroviário chinês. O espaço de estados utilizado no estudo de caso é relativamente pequeno, provando sua eficiência em modelos menores mas abrindo espaço para questionamentos sobre a eficiência do algoritmo

em modelos complexos. Inclusive essa é uma preocupação dos pesquisadores, que sugerem como um dos trabalhos futuros a aplicação da abordagem em modelos maiores e com mais funcionalidades. Wu e Schnieder [24] também consideram um subsistema de controle ferroviário via satélite a bordo como cenário de aplicação. Os autores apresentam uma abordagem de cima para baixo para gerar modelos CPN hierárquicos, de acordo com os diagramas de sequência da linguagem de modelagem unificada (UML). O espaço de estado do modelo CPN gerado é usado para derivar testes abstratos. Em modelos pequenos a abordagem se provou eficiente, porém o algoritmo utilizado na abordagem pode derivar sequências desnecessárias, além de apresentar desempenho reduzido em espaços de estados maiores. Esses são pontos que merecem atenção durante a execução da abordagem.

Liu *et al.* [25] aplicam simulações de modelo CPN para gerar testes de conformidade de um protocolo de rede. A abordagem proposta gera testes abstratos com base na teoria de teste do ioco (*i.e.*, uma abordagem de teste baseada no relacionamento entrada-saída) e considera o fluxo de controle dependente de dados dos comportamentos do sistema para derivar casos de teste viáveis. Durante a avaliação da abordagem, todos os requisitos do protocolo de rede foram cobertos. Além do modelo utilizado ser simples, a abordagem ioco precisa ser combinada com o modelo CPN para gerar os testes de conformidade corretamente.

Ruan *et al.* [26] apresenta um modelo CPN de um protocolo chamado *OpenFlow* e define um algoritmo para gerar testes abstratos com base no espaço de estados do modelo. Para reduzir o espaço de estado, o modelo CPN é analisado selecionando fichas e atribuindo os valores mais adequados para simplificar a modelagem. A avaliação do modelo *OpenFlow* é realizada para demonstrar a viabilidade da abordagem. Contudo, as informações do artigo são resumidas e fica pouco claro como aplicar a abordagem proposta.

Wang *et al.* [27] apresentam uma ferramenta chamada MBT/CPN para gerar testes abstratos a partir de modelos CPN, fornecendo suporte para gerar testes a partir de simulação ou espaço de estado. A ferramenta também fornece um recurso para exportar casos de teste como arquivos XML. Os autores se concentram na validação de protocolos de sistemas distribuídos. Um ponto negativo da ferramenta, é que os caminhos de teste não são apresentados, sendo necessária a análise dos casos de teste para extraí-los.

Cai *et al.* [29] apresentam uma abordagem para gerar testes abstratos usando o espaço de estados dos modelos CPN. Os autores descrevem a abordagem, juntamente com um estudo

de caso sobre uma máquina de venda automática de doces. A abordagem mostrou-se eficaz ao gerar a cobertura completa de estados, a cobertura de transições e a cobertura de pares de transições. Porém, o nível de complexidade do estudo de caso deixa dúvidas sobre a eficiência da abordagem em sistemas complexos.

Farooq *et al.* [30] descrevem uma abordagem para gerar testes abstratos a partir de modelos CPN traduzidos a partir de diagramas de sequência UML. Os autores geram testes abstratos considerando os critérios de cobertura durante a simulação do modelo CPN, com base no algoritmo de caminhada aleatória. Um sistema corporativo de negociação com clientes é usado como cenário de aplicação da abordagem. Uma das limitações apontadas é que o trabalho foca apenas no fluxo de controle de aspectos de um diagrama de atividades. Fluxos de dados ainda não fazem parte da abordagem.

Puspika *et al.* [31] aplicam uma variação do algoritmo DFS para derivar testes abstratos para aplicativos móveis usando o espaço de estados de modelos CPN. Os autores usam dois cenários de aplicativos: *Open GPS Tracker* e *Random Music Player*. Os modelos CPN são gerados a partir de classes UML e de diagramas de sequência UML. As seqüências geradas cobriram as propriedades de alcançabilidade, limitabilidade e vivacidade. O foco em um tipo de plataforma pode ser considerado como possível limitação.

Silva *et al.* [32] propõem uma abordagem baseada em simulações de modelos CPN para gerar testes abstratos para sistemas reativos cronometrados a partir de requisitos de linguagem natural. Os autores conduzem a análise da abordagem usando uma máquina de venda automática, um sistema de controle de usina nuclear, um sistema de controle de comando prioritário e um sistema de controle de indicador de mudança de direção, como cenários de aplicação. A análise se concentra em mostrar que a geração de casos de teste via simulação é viável e produz resultados promissores, especialmente quando os modelos possuem espaços de estado infinitos ou muito complexos. Como desvantagem, não há garantia de que os casos de teste abrangem todos os cenários descritos pelos requisitos, pois os testes abstratos são gerados por meio de simulações aleatórias de modelos CPN.

Jahan *et al.* [33] convertem processos de negócios originalmente representados usando a linguagem de execução de processos de negócios em um modelo CPN para testar propriedades específicas da composição de serviços web. A abordagem combina um grafo de alcançabilidade e um grafo de fluxo de controle para gerar casos de teste viáveis a partir

do espaço de estados. Com base no algoritmo DFS, os caminhos de teste devem obedecer a dois critérios estabelecidos: cobertura de estado e cobertura de transição. Um processo de aprovação de empréstimo é usado como cenário de aplicação para avaliar a abordagem. Algumas regras precisam ser seguidas para construir o grafo de fluxo de controle para então gerar os testes abstratos.

Kalamegan e Zayaraz [34] descrevem uma abordagem de priorização de teste abstratos. Um estudo de caso em um sistema de reserva de passagens aéreas também é usado para a composição de serviços web. Os autores pretendem aumentar a taxa de detecção de um conjunto de testes. Uma porcentagem média de detecção de falhas é usada para validar a eficiência da abordagem. O algoritmo denominado RDCT-G é aplicado com base em simulações de modelo para decidir as sequências a serem geradas. Os autores afirmam que os resultados são melhores do que as abordagens que utilizam algoritmos randômicos para este fim, porém, como é característico da abordagem por simulação, é possível obsercar que a porcentagem de cobertura é reduzida.

Kalamegam e Godandapani [35] propõem um algoritmo *on-the-fly* para gerar um conjunto de testes que abrange todos os caminhos possíveis, sem redundância para a composição de serviços web. Os autores também usam um estudo de caso sobre um sistema de reserva de passagens aéreas, considerando a priorização de testes abstratos, o tamanho do conjunto de testes e a redução de redundância. A abordagem utiliza simulação e concentra-se na cobertura de decisão, cobertura de entrada e saída e critérios de cobertura de transição. O algoritmo proposto oferece alta cobertura com o mínimo de esforço. Na abordagem, além do modelo CPN, o algoritmo necessita de pares de entrada e saída (UIO-pairs) como entrada para gerar corretamente os conjuntos de teste.

O estudo apresentado por Wang e Chen [36] descreve uma abordagem para a geração abstrata de testes para serviços web usando um modelo de processo de empréstimo CPN. Os autores propõem um algoritmo para gerar os testes usando a estrutura do modelo CPN (ou seja, arquivo XML). A saída do algoritmo proposto é uma equivalência de variáveis de predicado ou restrição. Assim como em outros estudos, o algoritmo de geração de testes abstratos necessita que a modelagem siga uma metodologia, nesse caso, um conjunto de modelos menores que representam unidades de programa executáveis.

Dong *et al.* [38] apresentam uma ferramenta para geração de testes abstratos a partir de

um modelo CPN que representa fluxos de trabalho de aplicações web. Os autores defendem a utilidade de garantir a racionalidade dos fluxos de trabalho desses aplicativos antes da fase de desenvolvimento do software. Como exemplo, um sistema de gerenciamento logístico é apresentado. A estrutura XML com informações sobre o espaço de estados extraída do modelo é utilizada como entrada para a ferramenta desenvolvida na linguagem Java, que os autores nomearam como AGTS. Uma vantagem apontada por eles é que a ferramenta é adequada para gerar sequências de teste para qualquer modelo CPN de forma automática. Entretanto, existe uma preocupação relacionada ao tamanho das sequências, principalmente em cenários de sistemas web, em virtude dos diversos fluxos de trabalho que compõem esse tipo de ambiente e que é o foco do trabalho. Este pode ser um problema em modelos com espaço de estados complexos.

Cai [37] propõe a geração de sequências e casos de teste para processos de negócio com base na composição de casos de teste. A linguagem BPEL é utilizada para descrever o comportamento dos processos e CPN para criar o que os autores chamam de redes de casos de teste. Um algoritmo para gerar os caminhos é proposto e os autores realizam a comparação dos resultados deste com o algoritmo de caminhada aleatória, obtendo resultados melhores e menos sequências redundantes. Porém é necessário seguir uma metodologia de modelagem que pode ser considerada complexa para que o algoritmo se comporte de forma esperada.

Wu e Krause [39] utilizam duas abordagens de testes baseados em modelos. Uma delas utiliza o grafico de alcançabilidade do modelo CPN, enquanto a outra utiliza uma variação de redes de petri chamada de SPENAT (Rede de Transição de Lugar Seguro com Atributos). Técnicas de geração baseadas em CPN e SPENAT são essencialmente semelhantes. No entanto, os autores afirmam que se o número de estados do sistema for grande e muitos estados forem simultâneos, a abordagem baseada no SPENAT é preferida. Para o exemplo do trabalho, um sistema de controle de trens via satélite, 24 caminhos são identificados no gráfico de alcançabilidade, que possui 66 nós e 84 arcos. O modelo de teste segue um padrão hierárquico de cenários que precisa ser obedecido na abordagem proposta, porém poucos detalhes são repassados sobre o processo de geração dos casos de teste.

Zhao *et al.* [40] apresentam um conjunto de regras de modelagem para tornar o modelo CPN mais adequado para a geração de testes, utilizando o cenário do sistema europeu de controle de trens. A abordagem proposta utiliza uma combinação de algoritmos e gera casos

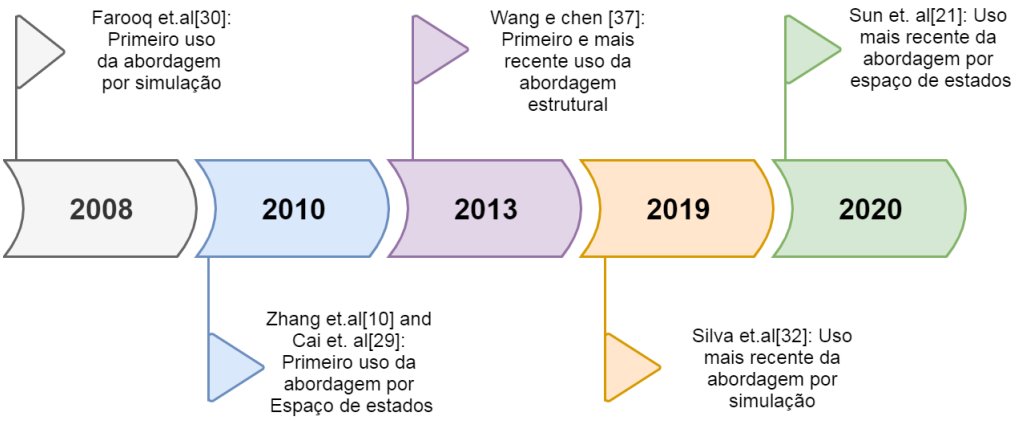


Figura 4.2: Linha do tempo das primeiras e recentes publicações por abordagem.

e sequências de teste em formato XML. Além disso, uma comparação breve é feita entre os modelos formais disponíveis para MBT, como Z e CSP. Os autores apresentam 12 regras de modelagem que devem ser obedecidas para aplicação da abordagem proposta, o que pode ser considerado uma desvantagem do ponto de vista prático, causando limtação em modelos existentes, por exemplo. Como vantagem, a abordagem facilita a automatização dos testes, fornecendo os casos e sequências de testes em formato XML, que pode ser utilizado para derivar os testes práticos com facilidade.

A Figura 4.2 apresenta uma linha do tempo das primeiras e recentes publicações para cada abordagem encontrada nos trabalhos, considerando a *string* de busca utilizada. A primeira publicação relacionada ao MBT foi em 2008, sendo o ano de 2010 o que mais teve publicações relacionadas ao tema. A publicação mais recente foi do ano de 2020 e desde a primeira publicação, apenas em 2009 não houve trabalhos sobre MBT, sugerindo que o tema continua em interesse pela comunidade nos últimos anos.

Na etapa de extração de dados, uma classificação para coletar informaçoes relevantes sobre as 26 publicações selecionadas na RSL. A descrição das informações coletadas contribuem para a qualidade da revisão e fornecem um panorama interessante sobre o tema MBT. A Figura 4.3 resume os principais problemas, soluções, propósitos e avaliação relatados nas 26 publicações durante a linha do tempo e uma breve discussão sobre esses tópicos é conduzida.

Propósito principal

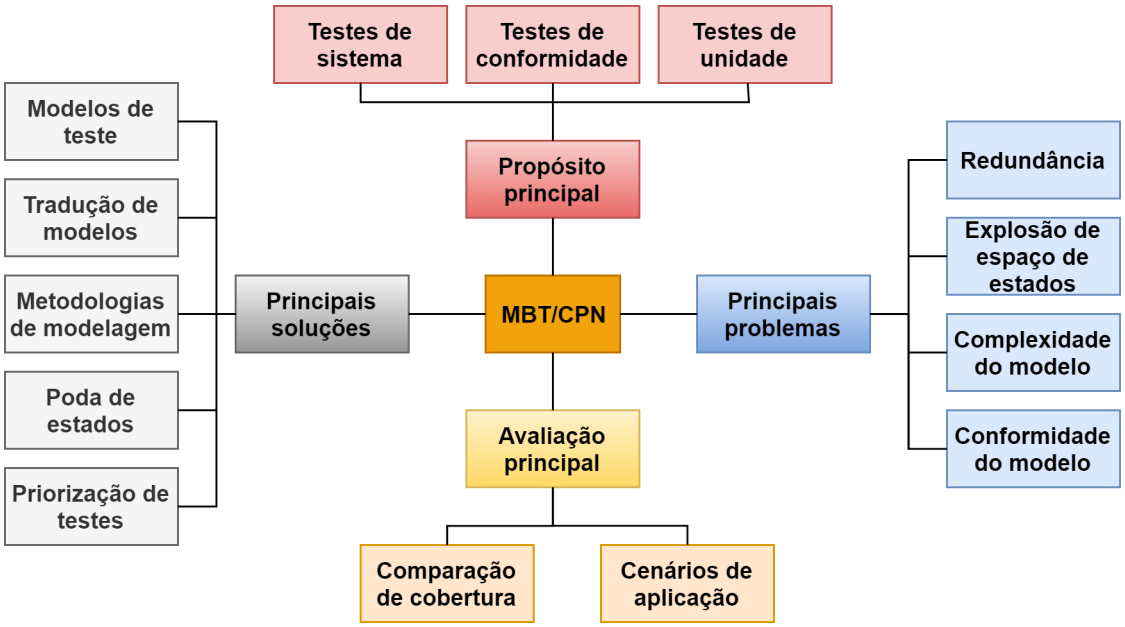


Figura 4.3: Principais problemas, soluções, objetivo e avaliação.

Essa classificação teve como objetivo identificar quais os principais objetivos de teste aplicando o MBT com modelos de CPN. As abordagens propostas geralmente visavam auxiliar os testadores durante os testes de sistema, testes de conformidade e teste de unidade. Os testes de sistema focam em testar o sistema como um todo, considerando cenários predefinidos, os testes de conformidade avaliam se o software foi desenvolvido de acordo com os padrões estabelecidos e os testes de unidade verificam os menores componentes de um sistema. Isso não significa que o MBT se limita unicamente a esses propósitos de teste, apenas que os trabalhos selecionados, quando abordam essa temática, apontam preferencialmente para eles.

Principais problemas

Na Figura 4.4 é possível visualizar uma árvore de níveis com os problemas encontrados durante o estudo. A complexidade e conformidade do modelo foram problemas frequentemente abordados pela definição de metodologias de modelagem e tradução de modelos, respectivamente. Estudos que oferecem metodologias de modelagem muito complexas podem ser considerados desvantajosos, pois a maior parte das soluções encontradas para aplicação de MBT sugere que uma metodologia de modelagem seja seguida, dificultando a utilização em modelos pré existentes, uma vez que modificar o modelo preservando o comportamento

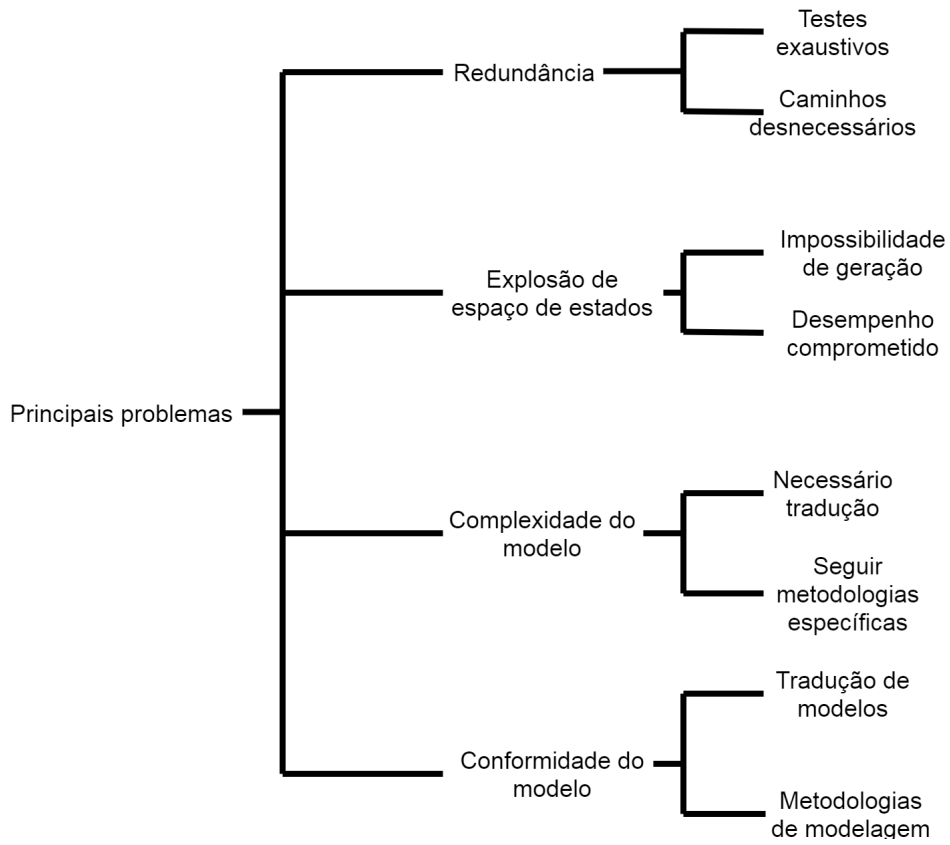


Figura 4.4: Árvore de níveis com os principais problemas encontrados na RSL.

original é um grande desafio. O mesmo acontece com a tradução de modelos que estão em linguagens ou formalismos diferentes, além disso, a tradução literal nem sempre é possível e enganos na tradução também podem ocorrer.

Dois outros problemas relevantes incluíram a redundância de testes abstratos e a explosão do espaço de estados. A redundância de testes é um problema que pode ocorer em qualquer uma das abordagens identificadas, podendo gerar testes exaustivos e desnecessários, já o problema de explosão de espaço de estados é característico de quando alguma abordagem baseada em algoritmos que manipulam o espaço de estados é escolhida.

Abordagens combinadas também podem sofrer com esse problema, caso utilizem exploração de espaço de estados em algum momento durante a execução. A explosão de espaço de estados é um problema crítico, pois quando não impede a geração dos testes abstratos, compromete fortemente o desempenho dos algoritmos. Os pesquisadores tentam contornar esses problemas com soluções que serão abordadas adiante.

Avaliação principal

Em geral, as abordagens propostas são avaliadas comparando a cobertura dos requisitos com outras implementações existentes, considerando um cenário de aplicação específico. Além disso, é comum encontrar comparações entre resultados de algoritmos originais e otimizados. Algoritmos tradicionais, como caminhada aleatória e DFS frequentemente são utilizados para fins de comparação com outros algoritmos propostos, sendo o DFS o mais citado.

Quantidade de testes e porcentagem de cobertura são utilizados para mensurar a qualidade das abordagens, e comparar por semelhança do cenário de aplicação ajudam os pesquisadores a validar os resultados.

Principais soluções

Na Figura 4.5 é apresentada a árove de soluções encontradas na RSL. Para evitar a explosão de espaço de estados que geralmente ocorre em modelos com complexidade elevada, a redução dos modelos de sistema e a poda de estados foram indicados como soluções para contornar o problema. A priorização de testes abstratos costuma ser indicada nos trabalhos da RSL como uma forma de reduzir a quantidade de testes desnecessários.

As metodologias de modelagem ajudam os testadores a criar modelos que possam ser aproveitados mais facilmente no momento de aplicar MBT para geração de testes abstratos. Modelos que desde o início são criados com base em metodologias específicas de modelagem costumam ter um ganho significativo de tempo quando é dado o momento de gerar os testes abstratos, uma vez que para adaptar modelos existentes que não seguiram tais metodologias se torna um trabalho demorado e, em alguns casos, pouco eficiente.

Em alguns casos, a tradução de modelos é necessária para aplicação de MBT. Ocasionalmente, os pesquisadores convertem diagramas da linguagem UML para modelos CPN para facilitar a condução da técnica e aproveitar informações previamente coletadas.

A partir da análise dos 26 artigos, foram identificadas informações relevantes sobre o uso de MBT com CPN, extraindo os dados definidos no protocolo de busca. A extração de dados permitiu responder às três questões de pesquisa.

Respondendo a QP 4

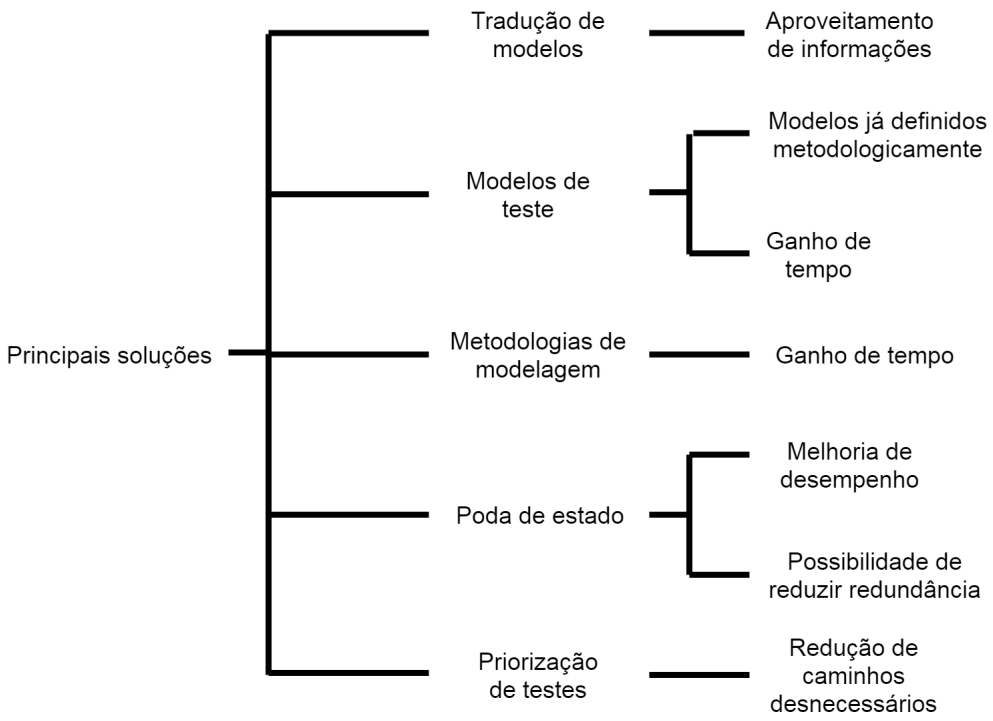


Figura 4.5: Árvore de níveis com os principais problemas encontrados na RSL.

A extração de dados dos 26 artigos permitiu a identificação de três abordagens diferentes para MBT usando CPN: simulação de modelo, análise de espaço de estado e análise estrutural. Cada estudo selecionou a abordagem de geração de testes abstratos com base nos cenários de aplicação e na complexidade do problema resolvido. Por exemplo, em [28, 32], os autores defendem o uso de simulação para gerar sequências, especialmente quando o espaço de estados do modelo é complexo ou infinito, com o objetivo de evitar o problema de explosão do espaço de estados.

A maioria dos estudos identificados manipula arquivos XML para ajudar na geração de testes concretos. Na Figura 4.6 são agrupados os trabalhos selecionados considerando as abordagens aplicadas. Atualmente, a simulação do modelo e a análise do espaço de estados são as abordagens mais usadas para gerar testes abstratos usando modelos CPN.

Respondendo a QP 5

O problema de explosão do espaço de estado é uma desvantagem crítica ao usar a abordagem do espaço de estado. Além disso, só é possível aplicar essa abordagem para analisar modelos de CPN com espaço de estado finito. Dependendo do tamanho do espaço de estado,

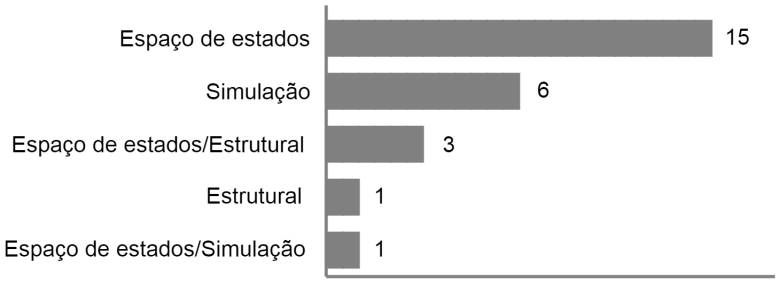


Figura 4.6: Número de artigos relacionados com uma abordagem específica.

os algoritmos podem demorar muito para explorar o grafo de alcançabilidade, exigindo o uso de uma técnica para reduzi-lo.

A abordagem de simulação é útil em cenários nos quais o modelo gerado possui espaço de estados muito grande ou infinito. No entanto, ao aplicar esta abordagem, existe o possível custo de derivar seqüências exaustivas, redundantes ou inúteis, dependendo de como as simulações de modelos CPN são realizadas. Também é relevante ressaltar que, ao usar simulação aleatória ou algoritmos como o de caminhada aleatória, não há garantia de cobertura completa dos requisitos [32]. A abordagem de espaço de estados lida com esse problema, melhorando a confiança na cobertura de estados e cobertura de transições, considerando os critérios de cobertura baseados em grafos.

Por fim, a abordagem estrutural se baseia na representação do modelo CPN. Os trabalhos que utilizam análise estrutural para derivar as sequências geralmente necessitam de ferramentas externas para auxiliar no processo, o que pode ser considerada uma desvantagem do ponto de vista prático, mas não interfere negativamente no resultado final. Dentre os trabalhos selecionados, é a técnica menos escolhida para fins de geração de testes abstratos, apesar de ser muito útil na construção dos oráculos e dos testes práticos.

Em geral, um número considerável de estudos exige que os modeladores sigam uma metodologia de modelagem específica para aplicar o MBT ao CPN. Se já existir um modelo de CPN, a abordagem exigiria adaptações no modelo de CPN para permitir o uso de MBT. Embora essas abordagens sejam válidas para especificar modelos de CPN com o objetivo principal de realizar o MBT, exigir que os modeladores sigam uma metodologia de modelagem é uma desvantagem crítica. Exigir adaptações em um modelo existente é ainda mais crítico, pois algumas adaptações podem ser extremamente difíceis de realizar, preservando os comportamentos do modelo. Essa desvantagem pode ocorrer com qualquer abordagem

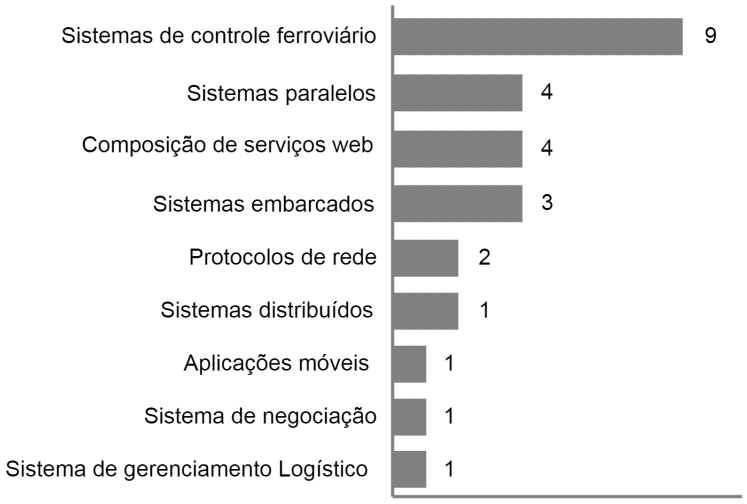


Figura 4.7: Número de artigos usando cada cenário de aplicação para a geração de testes abstratos.

específica que exija que os modeladores sigam uma metodologia de modelagem.

Respondendo a QP 6

Nos 26 trabalhos selecionados, não são apresentadas discussões sobre a qualidade dos testes abstratos gerados usando as três abordagens. Por exemplo, a verificação da cobertura dos requisitos de segurança de um sistema crítico específico para as três abordagens. Na Figura 4.7 é apresentado o número de artigos relacionados com cada cenário de aplicação. Apesar das abordagens serem aplicadas em diferentes cenários de aplicação, é evidente que a maioria dos estudos se concentra em sistemas críticos seguros, principalmente sistemas de controle ferroviário. Em nenhum dos estudos foram usados sistemas médicos como cenário de aplicação. Na maioria dos estudos também é aplicada a abordagem do espaço de estados, diferindo entre si pelos algoritmos implementados.

Os autores discutem frequentemente, de maneira geral, as vantagens e desvantagens do uso da abordagem utilizada. A escolha da abordagem geralmente depende do domínio de aplicação e das características do espaço de estado (*e.g.*, tamanho). No entanto, essa escolha também influencia na quantidade e qualidade dos testes abstratos gerados. A maioria dos estudos preocupa-se em garantir que todos os requisitos sejam atendidos (*i.e.*, todos os caminhos), sem considerar o quão crítico é o requisito a ser testado. Em alguns estudos, mais de uma sequência é gerada para o mesmo requisito, mas os autores não consideram

isso um problema em potencial ou apresentam uma discussão sobre prós e contras. Alguns estudos (*e.g.*, [18, 19, 35]) se concentram em abordagens para reduzir a quantidade gerada de seqüências inúteis ou redundantes, ajudando a melhorar a qualidade dos testes abstratos. Alguns algoritmos específicos são aplicados para gerar testes abstratos. Os algoritmos mais utilizados foram caminhada aleatória, DFS (incluindo variações) e *paths covered optimal*; enquanto o carteiro chinês, RDCTG e CTS-G aparecem com menos frequência.

Além de utilizar a extração de dados para responder as QP definidas, os trabalhos selecionados também foram classificados com base nos seguintes critérios:

- **Clareza (C1):** O nível de clareza durante a descrição da abordagem e dos resultados (*i.e.*, 1 para baixo, 2 para regular, 3 para bom e 4 para ótimo).
- **Relacionamento com sistemas críticos seguros (C2):** O nível de relacionamento do cenário de aplicação com sistemas críticos seguros (*i.e.*, 1 para não relacionado, 2 para de alguma forma relacionado e 3 para relacionado).
- **Completude (C3):** Nível de detalhes na apresentação dos algoritmos usados para implementar a abordagem (i.e., 1 para baixo, 2 para regular, 3 para bom e 4 para ótimo).

A classificação se fez necessária para nortear o estudo de caso deste trabalho. Com base nos critérios estabelecidos, os 26 trabalhos selecionados foram classificados e ordenados, com o objetivo de selecionar os 3 artigos com melhor classificação (considerando as três abordagens) a serem usados como base para o estudo de caso sobre sistemas médicos. Em caso de empate na classificação, o artigo foi selecionado com base na experiência da autora. Na Tabela 4.1 são apresentadas as classificações dos 26 trabalhos selecionados durante a RSL.

Tabela 4.1: Classificação dos 26 artigos selecionados durante a RSL.

Artigo	Cenário de Aplicação	Abordagem	C1	C2	C3	Total
[24]	Sistemas de controle ferroviário	Espaço de estados	4	3	3	10
[27]	Sistemas distribuidos	Espaço de estados/Simulação	3	2	4	9
[10]	Sistemas de controle ferroviário	Espaço de estados	4	3	2	9
[22]	Sistemas de controle ferroviário	Espaço de estados	2	3	3	8
[23]	Sistemas de controle ferroviário	Espaço de estados	3	3	2	8
[31]	Aplicações móveis	Espaço de estados	4	1	3	8
[7]	Sistemas de controle ferroviário	Espaço de estados/Estrutural	1	3	3	7
[19]	Sistemas paralelos	Espaço de estados	3	1	3	7
[33]	Composição de serviços web	Espaço de estados	3	1	3	7
[18]	Sistemas paralelos	Espaço de estados	3	1	3	7
[21]	Sistemas paralelos	Espaço de estados	3	1	3	7
[9]	Sistemas de controle ferroviário	Simulação	3	2	2	7
[35]	Composição de serviços web	Simulação	2	1	3	6
[39]	Sistemas de controle ferroviário	Espaço de estados	2	3	1	6
[30]	Sistema de negociação	Simulação	2	1	3	6
[32]	Sistemas embarcados	Simulação	2	2	2	6
[34]	Composição de serviços web	Simulação	2	1	3	6
[20]	Sistemas paralelos	Espaço de estados	3	1	2	6
[25]	Protocolo de rede	Simulação	3	1	2	6
[40]	Sistemas de controle ferroviário	Espaço de Estados/Estrutural	2	3	1	6
[8]	Sistemas de controle ferroviário	Espaço de estados/Estrutural	1	3	1	5
[36]	Composição de serviços web	Estrutural	2	1	2	5
[38]	Sistema de gerenciamento logístico	Espaço de Estados	2	1	2	5
[37]	Sistemas Embarcados	Espaço de estados	2	1	2	5
[29]	Sistemas Embarcados	Espaço de estados	2	1	1	4
[26]	Protocolo de Rede	Espaço de estados	2	1	1	3

(C1) Clareza; (C2) Relacionamento com sistemas críticos seguros; (C3) Completude.

4.2 Estudo de caso: Sistemas Médicos

A partir dos resultados da RSL, os trabalhos mais bem avaliados foram selecionados, considerando cada abordagem de geração de testes abstratos. Na Tabela 4.2 são descritos os trabalhos selecionados para o estudo, que será conduzido utilizando os modelos CPN de sistemas de ECG em versão reduzida(espaço de estado com 33 nós e 57 arcos) e completa (espaço de estados com 36 nós e 60 arcos) e de bomba de infusão de insulina reduzida (espaço de estados com 893 nós e 1133 arcos) e completa (espaço de estado com 6.317 nós e 16.426 arcos), ambos especificados em pesquisas anteriores [6, 15]. É importante dizer, que para a condução dos experimentos, a etapa de redução de modelos se caracteriza por priorizar o comportamento do software. Os modelos completos do modelo de bomba de

Tabela 4.2: Artigos selecionados na RSL para realizar o estudo de caso.

Título do Artigo	**Abordagem**	**Publicação**
Scenario-Based Modeling of the On-Board of a Satellite-Based Train Control System With Colored Petri Nets.	Análise de espaço de estados baseda em DFS.	IEEE Transactions on Intelligent Transportation Systems [24].
MBT/CPN: A Tool for Model-Based Software Testing of Distributed Systems Protocols Using Coloured Petri Nets.	Espaço de estados e Simulação.	International Conference on Verification and Evaluation of Computer and Communication Systems [27].
Automated Test Approach Based on All Paths Covered Optimal Algorithm and Sequence Priority Selected Algorithm.	Espaço de estados/Estrutural.	IEEE Transactions on Intelligent Transportation Systems [7].

infusão de insulina e do sistema de aquisição de ECG representam os comportamentos de hardware e software, enquanto nos modelos reduzidos, apenas o comportamento de software é representado. Essa redução foi realizada preservando o comportamento de software dos modelos e realizando as adaptações necessárias nas ligações entre os módulos. O passo a passo necessário para realizar essa redução está descrita a seguir:

1. Realizar a remoção de subpáginas que estão relacionadas com o comportamento de hardware;

2. Modificação da marcação de lugares para reduzir o espaço de estados;

3. Aplicar model checking (verificação de modelo) para analisar se as propriedades desejadas no modelo foram mantidas.

As alterações foram realizadas usando um laptop com 1,8 GHz Intel Core i5 Dual-Core. O uso dos dois sistemas médicos em diferentes representações teve como objetivo melhorar a análise empírica em relação ao tempo de execução para gerar os testes abstratos a partir de modelos com diferentes níveis de complexidade.

Na Figura 4.8(a) é ilustrada uma amostra do gráfico do espaço de estados extraído do módulo de software do modelo CPN do sistema de ECG descrito em pesquisa anteriormente publicada [6] (33 nós e 57 arcos). Os principais requisitos de software do ECG são a verificação da impedância dos eletrodos da pele e a situação da bateria. Na Figura 4.8(b) é ilustrada uma amostra do gráfico do espaço de estados extraído do módulo de software do modelo CPN do sistema de bomba de infusão de insulina descrito em [15] (893 nós e 1.133). Nesse caso, o principal requisito do software é o controle da infusão de insulina.

4.2.1 Abordagem por análise de Espaço de Estados Baseda em DFS

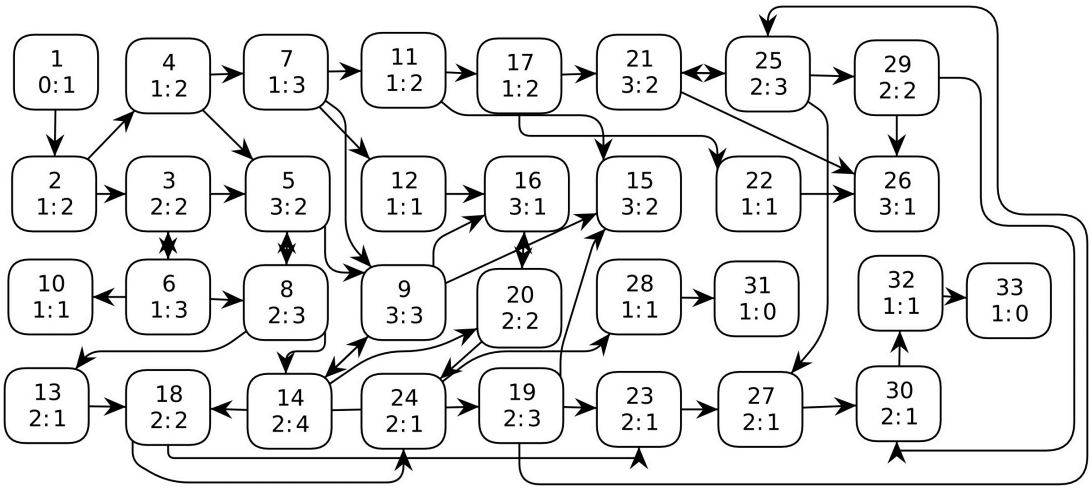
Wu and Schnieder [24] descreveram e aplicaram um algoritmo para gerar testes abstratos. Os caminhos gerados com o algoritmo estão relacionados com cenários possíveis ou a uma combinação de cenários de um sistema em teste, e o algoritmo extrai recursivamente todos os caminhos no grafo do espaço de estados, considerando os estados inicial e final. No presente trabalho, funções CPN/ML foram definidas para implementar o algoritmo proposto por Wu e Schnieder:

Algoritmo 1 Geração de caminho de teste

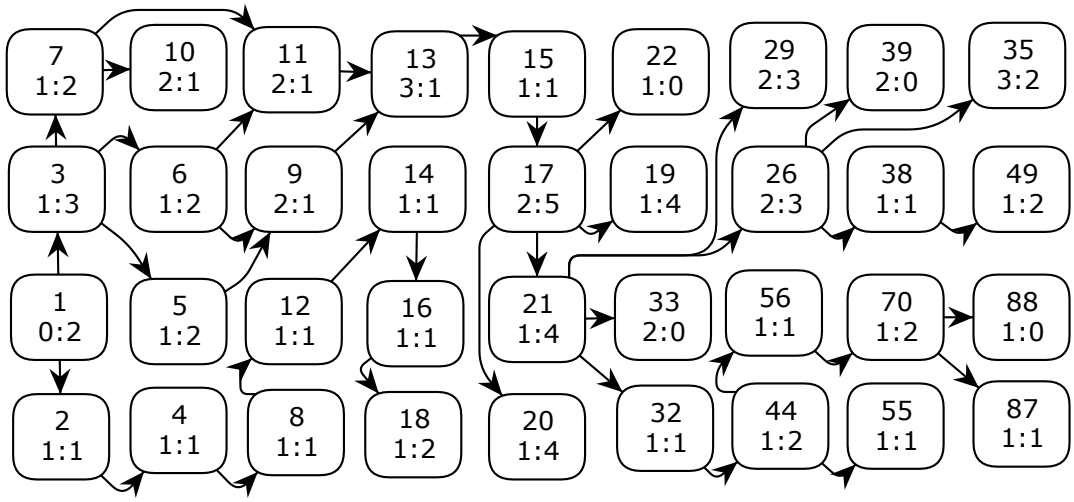
Entrada: espaço de estados

Saída: caminhos iniciando do estado inicial

```
1:  procedure CAMINHOS(init, final, caminho)
2:      visitado[init] = true
3:      para cada nó do grafo faça
4:          se ¬∃ ((caminho de init para atual) ∨ visitado) então
5:              Continue
6:              se atual = final então
7:                  Adiciona um caminho
8:              fim se
9:              CAMINHOS(atual, final, caminho + atual)
10:             visitado[atual] = false
11:         fim se
12:     fim para
13: fim procedure
```



(a)



(b)

Figura 4.8: (a) Gráfico completo do espaço de estado do módulo de software do modelo de ECG descrito em [6] (33 nós e 57 arcos). (b) Amostra do espaço de estados do módulo de software do modelo CPN dos sistemas de bomba de infusão de insulina descritos em [15] (893 nós e 1.133 arcos)).

O algoritmo foi desenvolvido separando funções principais e auxiliares. No total, cinco funcões em CPN/ML foram escritas para representar o pseudo-código acima, além de uma função específica para escrita em arquivo dos caminhos encontrados. As funções principais percorrem o grafo para encontrar os caminhos de teste, enquanto as funções auxiliares se preocupam com o armazenamento dos nós e a escrita dos caminhos em arquivo. O algoritmo lê como entrada o grafo referente ao espaço de estados do modelo e também seus estados

finais, para determinar um caminho entre o nó inicial e todos os estados finais do modelo. A implementação desse algoritmo e os arquivos necessários para sua utilização podem ser acessados online. [1].

O formato de entrada aceito pelo algoritmo para o grafo, é um arquivo em formato .sml denominado de *valgraph*. O arquivo é relativamente simples, constando apenas os pares de arestas extraídas do relatório de espaço de estados fornecido pela ferramenta CPN/Tools. Um segundo arquivo é necessário, contendo todos os estados finais do modelo em uso. Essa informação também pode ser extraída do relatório fornecido pela ferramenta.

Escritas as funções, elas podem ser acessadas pela ferrameta CPN/Tools utilizando a sintaxe de CPN/ML. Para execução do experimento, é necessário que no mesmo diretório, se encontre os arquivos *valgraph*, *valfinalState* e o arquivo que contém todas as funções necessárias, nesse experimento, denominado *newCPN*, além do arquivo do CPN/Tools que os utiliza, para produzir os resultados. Na Figura 4.9 são apresentados passos necessários no arquivo do CPN/Tools para executar o algoritmo.

```
use "newCPN.sml";
use "valGraphCPN.sml";
use "valFinalStateCPN.sml";

(*Encontra todos os caminhos para todos os estados finais*)
val AllPaths = GetAllPaths(graph,1,finalState);
val numAllPaths = length(AllPaths);

(*Fun para escrever no disco*)
use "writeFileCPN.sml";

(*Converte para String*)
val s_AllPaths = convertIntListList(AllPaths);
val s_numAllPaths = Int.toString numAllPaths;

(*Escreve no disco o resultado*)
writeFile "results_AllPaths.txt" s_AllPaths;
writeFile "results_numAllPaths.txt" s_numAllPaths;
```

Figura 4.9: Utilização do algoritmo dentro da ferramenta CPN/Tools.

Na Figura 4.10 é ilustrada uma amostra do grafo de espaço de estados extraído do modelo CPN completo (com representação de hardware e software) do sistema de bomba de infusão de insulina descrito por Costa *et al.* [15]. O grafo possui 6,317 nós, nos quais 444 dos nós representam estados finais. As linhas tracejadas foram adicionadas manualmente, e são

[1] https://https://github.com/illasilveira/experimentosdissertacao

usadas para representar os estados visitados para gerar um caminho entre os nós 1 e 220. Para cobrir todos os cenários, os testadores são solicitados a executar o algoritmo para todos os estados finais.

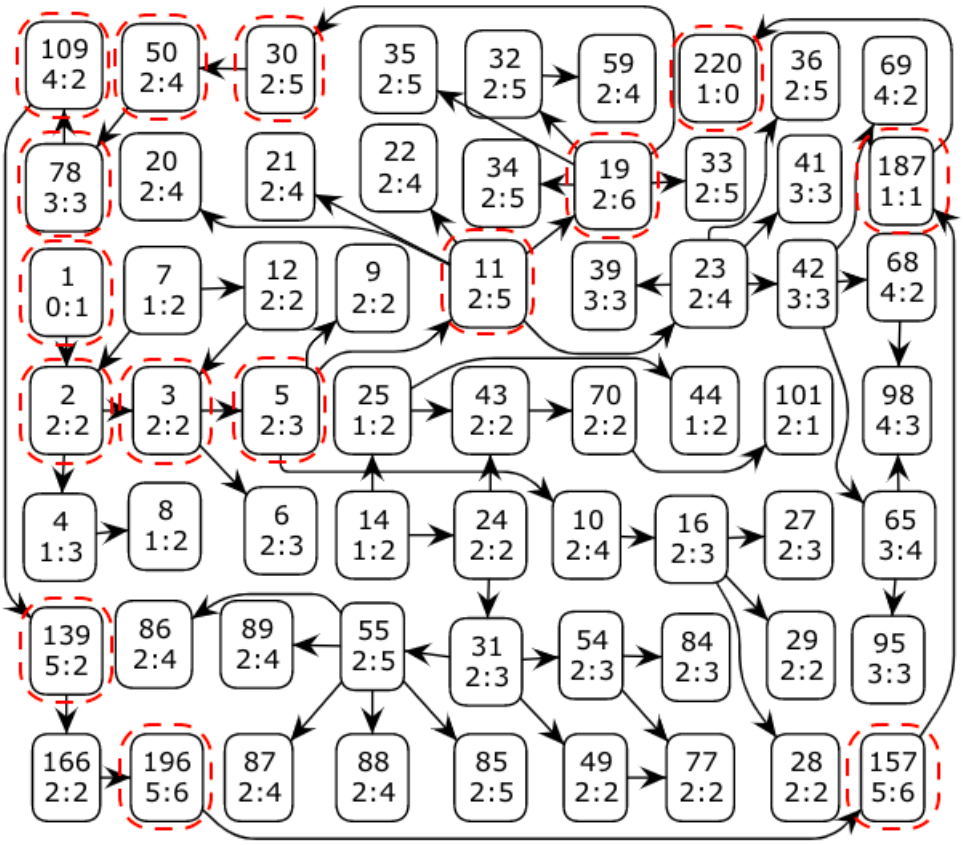


Figura 4.10: Amostra do grafo de espaço de estados para o modelo CPN de sistemas de bomba de infusão de insulina. As linhas tracejadas são usadas para representar os estados visitados para gerar um caminho entre os nós 1 e 220.

Para o modelo completo de CPN do sistema ECG (36 nós e 60 arcos), o algoritmo foi executado por menos de um segundo, para os dois estados finais do modelo e gerando 146 testes abstratos. Considerando o espaço de estado reduzido (Figura 4.8 a), que representa o módulo de software do sistema ECG, o algoritmo também gerou 146 testes abstratos com aproximadamente o mesmo tempo de execução. Analisando os 146 testes abstratos, um número considerável de caminhos desnecessários foram identificados. Por exemplo, é relevante verificar o posicionamento correto dos eletrodos e níveis adequados de bateria, pois implicam na medição correta (Figura 2.3). No entanto, para o sistema ECG não é relevante verificar este cenário considerando diferentes ordens de execução; por exemplo, o caminho

P1 = 1, 2, 3, 6, 10, 13, 18, 24, 28, 31 verifica primeiro a impedância dos eletrodos de pele, seguida pelo nível da bateria, e o caminho P2 = 1, 2, 4, 7, 12, 16, 20, 24, 28, 31 verifica o nível da bateria primeiro, seguido pela impedância dos eletrodos da pele. Portanto, apenas um subconjunto dos testes abstratos seria necessário para derivar testes concretos.

A execução da implementação CPN/ML do algoritmo de Wu e Schnieder por mais de 26 dias, usando o CPN/Tools e o modelo completo de CPN dos sistemas de bomba de infusão de insulina (espaço de estados com 6,317 nós e 16,426 arcos), nunca parou de executar e resultou em um número indefinido de testes abstratos. Além disso, considerando o módulo de software do modelo de bomba de infusão de insulina (espaço de estados com 893 nós e 1.133 arcos), o algoritmo rodou por aproximadamente 5 minutos, considerando os 223 existentes estados finais para gerar 38.300 testes abstratos. No módulo de software de infusão padrão (Figura 2.2), para a primeira tentativa de infusão, se ocorrer um erro crítico de software no momento exato da infusão, o disparo da transição *Erro Crítico 1* resulta em um dos caminhos possíveis relacionados ao estado final rotulado 88 (Figura 14 b). A transição de erro crítico 1 é ativada quando o usuário apenas seleciona a insulina basal e, para a próxima verificação de bateria, a carga total é 0 (zero). Este cenário refere-se a 6 dos 223 estados finais; No entanto, os testadores precisam apenas gerar um teste concreto de um dos 6 estados finais.

Com o objetivo de aumentar a confiança no algoritmo implementado, uma réplica do modelo CPN utilizado pelo artigo [24] foi desenvolvida a fim de verificar se os caminhos de teste gerados seriam semelhantes. Os resultados foram idênticos ao do artigo, com 67 caminhos gerados para 5 estados finais. O algoritmo levou menos de 1 segundo para gerar os testes abstratos. Os resultados desse experimento serviram apenas para atestar a correta implementação do algoritmo.

Para a condução do experimento foi utilizado um notebook com 6 GB de memória RAM e processador core i5 3210M com 2.5 Ghz de frequência. Todos os experimentos foram executados a partir da ferramenta CPN/Tools.

4.2.2 Abordagem por análise de Espaço de Estados/Simulação

Wang e Kristensen [27] propuseram uma ferramenta chamada MBT/CPN para auxiliar na geração de casos de teste usando modelos de CPN. A ferramenta foi importada usando SML no CPN/Tools e os algoritmos foram executados seguindo as instruções dos autores.

A ferramenta MBT/CPN está disponível gratuitamente on-line [2] junto com as instruções de uso. Para as abordagens de espaço de estado e simulação, a ferramenta requer a modificação de arquivos de configuração e a definição de funções SML específicas com base na seleção de transições específicas do modelo CPN onde a ferramenta será aplicada. As transições selecionadas devem estar relacionadas aos eventos de entrada e saída do sistema, afetando o número de testes gerados. Para aplicar a abordagem por simulação, a ferramenta também requer a definição de monitores para coleta de dados. Para o modelo de ECG, os monitores ilustrados na Figura 4.11 foram criados. Um monitor é um mecanismo disponível na ferramenta CPN/Tools utilizada para observar, inspecionar, controlar ou modificar a simulação de um modelo CPN. A ferramenta MBT/CPN disponibiliza um exemplo de como configurar os monitores, que observa os elementos de ligação do modelo durante a simulação.

```
▼SoftwareTC3
   Type: User defined
  ▼Nodes ordered by pages
   ▼Top
      Recharge (transition)
  ▼Init
     fun init () = ()
  ▼Predicate
     fun pred (bindelem) =
     let
       fun predBindElem (Top'Recharge (1, {recharge})) = true
         | predBindElem _ = false
     in
       predBindElem bindelem
     end
  ▼Observer
     fun obs (bindelem) =
     let
       fun obsBindElem (Top'Recharge (1, {recharge})) = [OutEvent (Rec(Recharge(1)))]
         | obsBindElem _ = []
     in
       SimConfig.observe(obsBindElem bindelem)
     end
  ▼Action
     fun action (observedval) = ()
  ▼Stop
     fun stop () = (SimConfig.stop())
```

Figura 4.11: Monitores definidos no CPN/Tools para geração de testes abstratos por simulação do modelo ECG.

Na Figura 4.12 são ilustradas as definições necessárias para a utilização do MBT/CPN no modelo. Basicamente, é preciso que uma variável que contém o caminho da ferramenta seja criada, além disso, colorsets que representam eventos de entrada e saída para gerar os casos de teste também são necessários. Na Figura 4.13 é apresentado como utilizar a biblioteca MBT/CPN na ferramenta CPN/Tools.

[2] https://github.com/selabhvl/mbtcpn

```
MBT
  val mbtcpnlibpath =
  "C:/Users/alvar/Documents/mbtcpn_v1.0/";
  colset TCInEvent = EVENTSH;
  colset TCOutEvent =  union Final:UNIT + Measure:UNIT + Rec:STATESS +
  Adj:STATESS;
  colset TCEvent = union InEvent : TCInEvent + OutEvent : TCOutEvent;
  use (mbtcpnlibpath^"config/simconfig.sml");
```

Figura 4.12: Definições necessárias para a utilização da ferramenta MBT/CPN.

```
use (mbtcpnlibpath^"build.sml");

val tcs = Execute.sim(100)
```

Figura 4.13: Como utilizar a ferramenta MBT/CPN no CPN/Tools.

Para o modelo completo de ECG, o MBT/CPN gerou 5 testes abstratos para criar 5 casos de teste concretos usando ambas abordagens, espaço de estados e simulação. Considerando o ECG com espaço de estados reduzido, a ferramenta foi executada por 5 segundos, gerando também 5 testes abstratos para gerar 5 casos de teste. A ferramenta também apresentou o mesmo resultado para a abordagem de simulação. Na Figura 4.14(a) é ilustrado o resultado da ferramenta usando o modelo de CPN completo e reduzido. Isto é razoável porque o MBT/CPN depende de entrada e saída de eventos, e que a apresentação da medida final, a notificação de bateria fraca e a notificação incorreta da impedância de eletrodos de pele são os eventos de saída exclusivos. Os requisitos relacionados ao status da bateria e eletrodos de pele e impedância representam a espera de estados que alertam o operador do sistema para corrigir configurações indesejadas. Dos testes gerados, apenas dois representam caminhos críticos de execução: (i) iniciar o sistema leva diretamente a uma medida de ECG; e (ii) iniciar o sistema, a bateria e os eletrodos da pele podem apresentar níveis indesejados que, após correções, levam a uma medida de ECG. No entanto, a suíte de testes incluía testes como por exemplo, (iii) iniciar o sistema, leva diretamente a níveis indesejados de bateria. Este requisito de teste já é considerado no caminho crítico (ii).

Para melhorar a discussão, a ferramenta MBT/CPN também foi aplicada para o modelo CPN de sistemas de bomba de infusão de insulina. Por um lado, usando o CPN/Tools e o espaço de estados do modelo completo de CPN, o MBT/CPN foi executado por 1 segundo, gerando 8 testes abstratos para criar o mesmo número de casos de teste concretos. Na Fi-

gura 4.14(b) ilustra o resultado da ferramenta usando o modelo CPN completo, no qual os eventos de saída *si* e *pi* representam infusões padrão e personalizadas; enquanto os erros (por exemplo, *OutEvent error 1*) significam falhas críticas no software em etapas específicas dos modos de infusão. A ferramenta foi executada por 7 segundos e usou o mesmo número de testes abstratos para a abordagem de simulação. Por outro lado, usando a abordagem de simulação para o modelo CPN considerando somente componentes de software, a ferramenta trabalhou por 5 segundos, também gerando 8 testes abstratos. Para a abordagem por espaço de estados, a ferramenta foi executada por 1 segundo e apresentou o mesmo número de testes abstratos. Na Figura 4.14(c) é ilustrado o resultado da ferramenta usando o modelo CPN reduzido. Portanto, o tamanho do modelo de CPN não afetou o número de casos de teste gerados usando a ferramenta MBT/CPN. Para esses modelos, focando em caminhos críticos de execução, o mesmo requisito de teste está incluído em casos de teste diferentes (por exemplo, erro crítico 1).

4.2.3 Abordagem por Espaço de estados/Estrutural

Zheng *et al.* [7] usaram os algoritmos APCO e SPS para conduzir o MBT. Como requerido na metodologia proposta, as estruturas dos modelos CPN foram analisadas, para identificar os nós que representam as portas de entrada e saída de subpáginas que estão iniciando e terminando cada cenário de teste. A identificação foi conduzida a partir da simulação dos modelos em conjunto com o acesso ao espaço de estados gerado previamente, para identificar o nó correspondente à porta em uso. Cada requisito foi simulado afim de realizar o levantamento dos pares de entrada e saída necessários para a utilização do algoritmo de forma correta. Funções CPN/ML foram definidas para representar os algoritmos usados e gerar testes abstratos a partir do espaço de estados, guiados pelas portas de entrada e saída identificadas. Os espaços de estados de cada sistema médico foram reduzidos usando os componentes fortemente conectados, seguindo as etapas metodológicas necessárias para aplicar os algoritmos.

A proposta dos autores é fornecer um método de otimização de caminho, utilizando a combinação dos dois algoritmos propostos. Enquanto o algoritmo APCO é aplicado para gerar casos de teste, o SPS foca em gerar os testes abstratos. Considerando que os autores consideram o algoritmo DFS problemático, o algoritmo APCO surge como uma versão me-

```
val tcs =
 [[InEvent (Button ()),OutEvent (Rec (Recharge 1)),OutEvent (Measure ())],
  [InEvent (Button ()),OutEvent (Rec (Recharge 1)),OutEvent (Adj (Adjust 5)),
   OutEvent (Measure ())],[InEvent (Button ()),OutEvent (Adj (Adjust 5))],
  [InEvent (Button ()),OutEvent (Measure ())],
  [InEvent (Button ()),OutEvent (Adj (Adjust 5)),OutEvent (Measure ())]]
 : TCEvent list list
```

(a)

```
val tcs =
 [[InEvent 1],[InEvent 1,OutEvent error3],[InEvent 1,OutEvent error4],
  [InEvent 1,OutEvent error2],[InEvent 1,OutEvent si],
  [InEvent 1,OutEvent si,OutEvent error1],
  [InEvent 1,OutEvent si,OutEvent error2],
  [InEvent 1,OutEvent pi,OutEvent error4],[InEvent 1,OutEvent pi],
  [InEvent 1,OutEvent pi,OutEvent error3],[InEvent 1,OutEvent error1]]
 : TCEvent list list
```

(b)

```
val tcs =
 [[InEvent 1,OutEvent si],[InEvent 1,OutEvent pi],[InEvent 1],
  [InEvent 1,OutEvent error1],[InEvent 1,OutEvent error6],
  [InEvent 1,OutEvent error3],[InEvent 1,OutEvent error4],
  [InEvent 1,OutEvent error5],[InEvent 1,OutEvent error2]]
 : TCEvent list list
```

(c)

Figura 4.14: (a) Resultados da ferramenta MBT/CPN por espaço de estados/simulação para o modelo completo e reduzido do ECG (b) Resultados da ferramenta MBT/CPN por espaço de estados/simulação para o modelo completo da bomba de infusão de insulina. (c) Resultados da ferramenta MBT/CPN por espaço de estados/simulação para o modelo reduzido da bomba de infusão de insulina.

lhorada do DFS, pesquisando todos os caminhos possíveis entre qualquer nó do conjunto de estados iniciais e o nó correspondente no conjunto de nós de estados finais, e em seguida, remove os caminhos redundantes. Dessa forma, o conjunto de caminhos mais simplificado que cubra todos os caminhos é escolhido [7].

O Algoritmo SPS manipula os cenários dos modelos para gerar sequências e subsequências de teste a partir das portas de entrada e saída dos cenários. As subsequências são geradas para cada cenário e atribuindo pesos para cada subsequência encontrada, as sequências de teste são montadas do cenário inicial ao final. Em seguida, sequencias redundantes são efetivamente reduzidas. O pseudo-código dos algoritmos APCO e SPS estão listados no Algoritmo 1 e Algoritmo 2, respectivamente.

Para conduzir o experimento com os algoritmos, a utilização do grafo do modelo em questão é necessário. Os grafos são previamente extraídos do espaço de estados do modelo

Algoritmo 2 Algoritmo APCO - All paths covered optimal

```
1:  procedure PROFUNIDADE(nós, nós passados)
2:      Nó extendido; matriz de subnós;
3:      para i = 1 para tamanho(SN) faça
4:          NP do SN[i] = + SN[i];
5:      fim para
6:      Saída iniciada = Falso; Saída profundidade = Falso;
7:      para i = 1 ao tamanho(SN) faça
8:          se SN[i] = G então Saída PN de SN[i]; Saída = Verdadeiro;
9:          senão se SN[i] não está em nós desabilitados ou nós passados então
10:             Saida profundidade = Profundidade(SN[i], NP do SN[i]);
11:         fim se
12:     fim para
13:     Resultado = Saída ou Saída de profundidade;
14:     se Resultado = Falso então Inserir nó em ND;
15:     fim se
16:     Retorna Resultado;
17: fim procedure
```

Algoritmo 3 Algoritmo SPS - Sequence priority selected

```
1:  procedure MAX UNIT(dado 1, dado 2)
2:      para i = 0 para numero total de cenários faça
3:          dado1[i]=numero de subsequencia do cenario i
4:          dado2[i]=numero de subsequencia do cenario i
5:          para j=0 para dado1[i].tamanho faça
6:              Encontre max dado2[i]; id=j;
7:              Retorne ret=dado1[i].p[j];
8:              Apague dado1[i].p[j]
9:              Saída cenário i para maxdado2[i] e dado1[i].p[id]
10:         fim para
11:         Chamada de função (sequencia, cenário,tamanho)
12:         Rearranja cenário, do maior para o menor;
13:     fim para
14:     Retorna cenário, tamanho;
15:     Saída sequência;
16: fim procedure
```

CPN. Além disso, os pares de entrada e saída dos cenários dos modelos foram identificados. Por uma questão de limitação do algoritmo, os testes abstratos não podem ser gerados de uma única vez, sendo necessária a execução de cada requisito e seus pares de portas correspondentes por vez. Por exemplo, O modelo de bomba de infusão de insulina possui 12 requisitos que levam a estados finais, sendo o restante dos estados finais do modelo uma variação desses 12. Desse modo, o algoritmo foi executado 12 vezes, uma para cada conjunto de portas que representa um requisito. Essas informações foram inseridas num arquivo .txt, que continha o grafo do modelo e seus respectivos pares de entrada e saída dos cenários.

Uma vez que o arquivo de entrada foi configurado, é necessário utilizar as funções dentro do CPN/Tools. Tanto os arquivos dos algoritmos, quanto as variáveis de entrada precisam estar no mesmo diretório do arquivo do CPN/Tools. Os algoritmos implementados e os arquivos necessários para sua utilização estão disponíveis online [3]. Na Figura 4.15 é apresentado como utilizar os algoritmos na ferramenta.

```
(*First*)
use "vals.sml";
use "APCO_CPN.sml";
use "SPS_CPN.sml";


(*Second*)
val showMapTestCases = mapTestCases(S,testCases);
val testSequences = insertETS(STS,SS,ETS);
val lenTestSequences = List.length(testSequences);
```

Figura 4.15: Utilização dos algoritmos APCO e SPS no CPN/Tools.

Para os modelos completo e reduzido de CPN do sistema de ECG, os algoritmos foram executados por 2 segundos, gerando 4 testes abstratos para os dois modelos. A partir dos testes gerados, é necessário considerar o significado de caminhos específicos. Por exemplo, existem dois caminhos que representam os valores corretos de bateria e eletrodos de pele impedância antes da medição, na qual apenas um seria necessário para gerar testes concretos. Dependendo do modelo CPN especificado, o número de testes desnecessários ou inviáveis aumentam. Os autores afirmam a necessidade de algumas regras de modelagem para especificar os sistemas; no entanto, as regras não são apresentadas claramente.

Usando o CPN/Tools e o modelo completo de CPN da bomba de infusão de insulina,

[3] https://github.com/illasilveira/experimentosdissertacao

mesmo aguardando 7 dias, o algoritmo nunca parou de trabalhar e resultou em um número indefinido de testes abstratos. Um problema semelhante aconteceu ao usar o algoritmo proposto por Wu e Schnieder [24] no mesmo modelo de CPN, devido ao problema de explosão do espaço de estados. Este problema foi resolvido considerando o modelo de CPN reduzido. O algoritmo de Zheng *et al.* [7] trabalhou por 12 segundos para gerar 54 testes abstratos. Quando comparado com os resultados do algoritmo de Wu e Schnieder [24], o número de testes abstratos diminuiu; no entanto, ainda é possível identificar testes abstratos desnecessários.

4.3 Discussões

Considerando os trabalhos e abordagens idenfificadas na RSL, foi possível realizar algumas análises com relação a qualidade dos testes abstratos gerados e sua cobertura em relação ao sistema sob teste. Os trabalhos que utilizam a abordagem de espaço de estados e algoritmos baseados em DFS costumam entregar cobertura de todos os estados e transições. Exemplos de trabalhos que utilizam essa abordagem com cobertura total são [33] e [24].Nos dois trabalhos, os critérios de teste estabelecidos foram contemplados. É perceptível que as abordagens que escolhem esse algoritmo conseguem alta cobertura de estados e de transições, em virtude da natureza dele. Mas testes abstratos redundantes podem existir, por exemplo, para uma mesma funcionalidade, demandando por vezes adaptações na técnica e/ou nos algoritmos. Um exemplo dessa prática pode ser vista no trabalho de [18] que apresenta um algoritmo para seleção de testes abstratos. Essa é uma das formas de contornar o problema da redundância.

Trabalhos que adaptam o algoritmo DFS podem ser vistos em [22] e [31]. Uma comparação entre o algoritmo adaptado em [22] com o DFS padrão demonstra que o algoritmo adaptado entrega melhor performance, mais casos de teste e maior cobertura para o cenário proposto. Mas em virtude das regras de modelagem da abordagem proposta, pode-se dizer que os resultados são derivados de um conjunto de fatores, como a adaptação do algoritmo e a metodologia de modelagem aplicada. Em [31] a adaptação proposta cobre todos os estados e transições, mas os autores não dão informações sobre redundância ou qualidade dos testes abstratos extraídos, porém é possível perceber nos resultados de um dos modelos utilizados

que sequências redundantes estão presentes.

A poda de estado também se mostra eficiente em modelos complexos que utilizam espaço de estados, o trabalho de [19] apresenta uma técnica de poda de estado que visa a cobertura total para todos os comportamentos em teste sem redundância.

De modo geral, abordagens que utilizam espaço de estados combinados com algoritmos, conseguem entregar cobertura de todos os requisitos. Algoritmos como DFS e suas variações costumam gerar testes abstratos com alta cobertura. Sem adaptações, porém, o número de testes asbtratos gerados pode ser alto e redundante, por esse motivo alguns trabalhos focam em reduzir a redundância adaptando o comportamento do DFS ou criando novos algoritmos e abordagens para melhorar a qualidade dos testes. Em modelos com espaço de estados muito grandes a demora na execução do algoritmo também pode ser um problema.

Para os testes abstratos gerados a partir de simulação, alguns algoritmos tradicionais também podem ser encontrados nas abordagens, como é o caso do algoritmo do carteiro chinês no trabalho de [9] e caminhada aleatória em [11]. Ambos necessitam de moetodologias de modelagem específicas para aplicação da abordagem. Em [9] os autores conseguem atingir cobertura completa para o estudo de caso proposto. Em [11] algumas análises de qualidade e cobertura de teste são realizadas. O conjunto de testes gerado é considerado adequado para atender aos critérios de cobertura sequencial, mas insuficiente para atender a critérios de comportamento simultâneo. Além disso, mesmo com o algoritmo proposto, que utiliza caminhada aleatória com algumas adaptações, a cobertura é baixa em alguns cenários e redundâncias são encontradas com frequência. Os autores sugerem melhorias no algoritmo para trabalhos futuros.

Alguns trabalhos apresentam opçoes de algoritmos que otimizam os resultados da geração de testes abstratos. Um exemplo pode ser visto no trabalho de [34]. O algoritmo RDCTG proposto, além de gerar testes abstratos, possui um mecanismo de priorização de sequências. Uma métrica de porcentagem de detecção de falhas é utilizada para demonstrar que o algoritmo RDCTG possui uma taxa de 67 porcento de detecção de falhas contra 17 porcento dos algoritmos randômicos. Em [35] o algoritmo CTS-G é proposto para gerar testes abstratos mediante algumas entradas específicas, como pares únicos de entrada e saída (UIO-pairs) e o modelo CPN. A qualidade dos resultados é analisada com base em 4 critérios de cobertura. Os autores afirmam que o algoritmo proposto é capaz de lidar com redundâncias e prioriza-

ção de testes, e que oferecem alta cobertura com o mínimo de esforço. Para o estudo de caso do trabalho, a cobertura foi total.

A maior vantagem da abordagem por simulação em comparação com espaço de estados é a facilidade em contornar o problema de explosão de espaço de estados e loops infinitos. Por esse motivo, os trabalhos indicam que essa é a opção mais viável em sistemas de alta complexidade ou que possuem espaço de estados infinito. Por outro lado, alguns algoritmos aplicados à simulação também podem entregar testes abstratos redundantes e a cobertura total se torna mais difícil de ser alcançada com essa abordagem.

Os trabalhos que utilizam a abordagem estrutural tem como característica a utilização do arquivo XML do modelo CPN para realizar a leitura dos dados estruturais. Além disso, uma implementação externa geralmente é necessária para realizar a leitura. No trabalho [38] os autores desenvolveram uma ferramenta chamada AGTS que como entrada lê o XML do modelo CPN e como saída gera testes abstratos. Nesse caso, o grafo de alcançabilidade é utilizado para guiar a geração dos caminhos. Os critérios de cobertura escolhidos foram cobertura de arcos e cobertura de estados. Ambos foram atingidos com a ferramenta e estudo de caso apresentados. Em [7] o algoritmo APCO, que apresenta melhorias no algoritmo DFS padrão, é aplicado ao XML para extrair sequências, sub-sequências e casos de teste baseados em dois critérios de cobertura: Todos os nós e todas as ramificações, e os resultados mostram que a cobertura foi atingida. Em comparação com o DFS padrão, o algoritmo APCO obteve melhores índices de cobertura.

A maior parte dos trabalhos que aplicam a abordagem estrutural selecionados, apesar de escolherem o arquivo XML para ser a entrada da geração dos caminhos, preferem utilizar a estrutura do espaço de estados para mapear o modelo. A diferença entre eles geralmente é o algoritmo utilizado para percorrer o grafo e as regras de modelagem. O que sugere que a cobertura dos testes gerados a partir da abordagem estrutural é semelhante à cobertura por espaço de estados, uma vez que a principal diferença encontrada entre as duas abordagens é a maneira com que os testes abstratos são gerados.

Nenhum dos estudos selecionados na RSL discutiu a relação entre as três abordagens abstratas de geração de teste, são elas: espaço de estados, simulação e estrutural. Em um pequeno número estudos, alguns autores apresentam breves discussões sobre vantagens e desvantagens, considerando apenas as abordagens de espaço de estados e simulação. A falta

de comparações abrangentes das abordagens abstratas de geração de testes para CPN limita as contribuições de discussões existentes. A RSL e o estudo de caso se complementaram para melhorar essas comparações.

Segundo os resultados da RSL, as implementações da abordagem por espaço de estados contam com algoritmos baseados em pesquisa, visando maximizar a cobertura de requisitos usando o gráfico de alcançabilidade do espaço de estados. Para obter alta cobertura, os algoritmos geralmente exigem uma pesquisa exaustiva do espaço de estados. No entanto, apesar de ser uma vantagem, alta cobertura também pode resultar em testes abstratos redundantes, levando a casos de teste concretos desnecessários. Em alguns casos, as implementações abordam esse problema priorizando e selecionando caminhos específicos do gráfico, enquanto, em outros casos, as implementações requerem a redução de uma representação completa de um sistema para gerar um modelo de teste. Nos dois casos, o modeladores precisam seguir alguma metodologia de modelagem para conseguir reutilizar os modelos de CPN para teste; caso contrário, o uso requer adaptações nos modelos existentes. A desvantagem de priorizar/selecionar caminhos ou gerar um modelo de teste é o trabalho extra necessário, que pode ser demorado e exigir conhecimento especializado e detalhado sobre o modelo. Por exemplo, o CPN/Tools não fornece contra-exemplos de verificação de modelo, exigindo codificação para estender as funcionalidades. Tal cenário pode ser ainda mais problemático quando a equipe responsável pela modelagem não é a mesma responsável pela reutilização do modelo para teste. Outra desvantagem ocorre quando o espaço de estados é infinito, porque o a manipulação de espaços de estados parcial compromete a cobertura de requisitos.

Para a geração de testes abstratos baseados em simulação, implementações específicas dependem de algoritmos como o do carteiro chinês [9] e caminhada aleatória [30]. As implementações geralmente exigem que os modeladores especifiquem modelos de CPN seguindo metodologias de modelagem. Além disso, a cobertura de requisitos pode ser baixa e apresentar testes redundantes para alguns cenários, dependendo do algoritmo implementado. Um exemplo de vantagem da abordagem de simulação, quando comparada à abordagem por espaço de estados, é a possibilidade de lidar com espaço de estados infinito e a ocorrência de loops infinitos. Portanto, os estudos selecionados na RSL indicam que a abordagem de simulação é adequada quando os testadores precisam aplicar o MBT para sistemas complexos que tem espaço de estados muito grande ou infinito. No entanto, a cobertura total de

requisitos de teste é dificil de alcançar.

A abordagem estrutural baseia-se na análise de arquivos XML que representam modelos de CPN, exigindo trabalho extra. Por exemplo, um O grafo gerado a partir do modelo CPN pode ser usado junto com algoritmos de busca para orientar a geração de testes abstratos. Analisando os estudos selecionados na RSL, observamos que a cobertura dos requisitos de teste por implementações do abordagem estrutural pode ser semelhante à cobertura obtida por implementações da abordagem por espaço de estados, dependendo do grafo gerado a partir da estrutura do modelo. A vantagem é que a abordagem estrutural lida com modelos que apresentam espaço de estados infinito, ao custo de implementações mais complexas. No entanto, o trabalho relacionado à abordagem estrutural [36] apresenta baixos níveis de clareza e completude, o tornando inadequado para o estudo de caso e discussões detalhadas.

Na Tabela 4.3 são resumidos os resultados do estudo de caso, realizado para complementar a RSL. Para os modelos de ECG e bomba de infusão de insulina, em geral, a implementação do algoritmo de Wu e Schnieder [24] geraram o maior número de testes abstratos, usando a abordagem por espaço de estados. Para o modelo bomba de infusão de insulina completo, o algoritmo gerou um número indefinido de testes abstratos devido ao problema de explosão de espaço de estados. Com base em espaço de estados e simulação, a ferramenta MBT/CPN [27] apresentou melhores resultados e impediu o problema de explosão de espaço de estados, ao custo de diminuir a cobertura de requisitos menos críticos. Tal situação ocorre porque a ferramenta MBT/CPN requer que os testadores definam entrada e saída eventos, aumentando os riscos de definições enganosas ou perda de informação. Analisando o arquivo XML dos modelos de CPN e espaço de estados, as implementações dos algoritmos apresentados por Zheng *et al.* [7] geraram mais testes abstratos quando comparados à ferramenta MBT/CPN, com a vantagem de simplificar o processo de geração de oráculos de teste e testes concretos. O algoritmo também gerou um número indefinido de testes abstratos para o modelo de bomba de infusão de insulina completo devido ao problema de explosão de espaço de estados.

Os resultados obtidos através da redução dos modelos no estudo de caso mostraram que ainda que os modelos tenham sido reduzidos, uma boa cobertura pode ser obtida em sistemas com modelos complexos, com o benefício de melhorar o desempenho dos algoritmos. Essa estratégia pode auxiliar testadores que enfrentam os problemas clássicos da utilização de

algoritmos baseados em grafos.

Tabela 4.3: Resumo dos resultados do estudo de caso.

Id	Abordagem	Modelo	Testes Abstratos	Requisitos
[24]	Espaço de estados	ECG completo	146	Todos os críticos
[24]	Espaço de estados	ECG reduzido	146	Todos os críticos
[24]	Espaço de estados	Bomba completa	Explosão de estados	Explosão de estados
[24]	Espaço de estados	Bomba reduzida	38.300	Todos os críticos
[27]	Espaço de estados	ECG completo	5	Todos os críticos
[27]	Espaço de estados	ECG reduzido	5	Todos os críticos
[27]	Espaço de estados	Bomba completa	8	Todos os críticos
[27]	Espaço de estados	Bomba reduzida	8	Todos os críticos
[27]	Simulação	ECG completo	5	Todos os críticos
[27]	Simulação	ECG reduzido	5	Todos os críticos
[27]	Simulação	Bomba completa	8	Todos os críticos
[27]	Simulação	Bomba reduzida	8	Todos os críticos
[7]	Espaço de estados/Estrutural	ECG completo	4	Todos os críticos
[7]	Espaço de estados/Estrutural	ECG reduzido	4	Todos os críticos
[7]	Espaço de estados/Estrutural	Bomba completa	Explosão de estados	Explosão de estados
[7]	Espaço de estados/Estrutural	Bomba reduzida	54	Todos os críticos

No entanto, a comparação empírica apresenta algumas limitações à validade. Por um lado, as *strings* de busca podem não cobrir todos os trabalhos existentes. Para contornar esse problema, uma pesquisa manual e a técnica *snowballing* foram realizadas.Por outro lado, no estudo de caso, utilizar apenas três das implementações podem limitar a discussão. Esse problema foi abordado concentrando-se nas características gerais dos 26 trabalhos e suas implementações específicas. As 16 implementações baseadas em espaço de estados usaram

algoritmos baseados em pesquisa. As 6 implementações baseadas em simulação usaram algoritmos para guiar as simulações, enquanto as 4 implementações baseadas em estruturas exigiam um analisador XML.

Capítulo 5

Conclusões e trabalhos futuros

Dados os resultados da revisão terciária da literatura, uma lacuna de pesquisa foi identificada. Embora redes de Petri coloridas (*Coloured Petri Nets* - CPN) tenha grande potencial para aplicação de teste baseado em modelos, os estudos não consideraram o formalismo em suas revisões. Além disso, foi possível notar na revisão sistemática que não haviam trabalhos que oferecessem aos pesquisadores uma análise comparativa de abordagens disponíveis para aplicação de teste baseado em modelos com CPN.

A partir da Revisão Sistemática da Literatura (RSL), três abordagens gerais de geração de teste abstratos foram identificadas e estão disponíveis para testadores que reutilizam modelos de CPN: espaço de estados, simulação eestrutural. Os resultados da análise empírica mostraram que a escolha de uma abordagem geralmente depende do tamanho do espaço de estados, que está ligado à complexidade do modelo.

Nos experimentos conduzidos com a abordagem de espaço de estados e o algoritmo DFS, foi possível observar que o modelo completo da bomba de infusão de insulina não obteve resultados satisfatórios. Por outro lado, nos modelos de bomba de infusão de insulina reduzida e ECG, caracterizados por terem menor complexidade, o algoritmo conseguiu entregar os testes abstratos sem maiores dificuldades. Porém, devido à característica do algoritmo DFS, testes abstratos redundantes puderam ser identificados nos resultados.

Utilizando a abordagem por simulação, a geração dos testes abstratos para os quatro modelos ocorreu de forma rápida, mesmo para o modelo completo da bomba de infusão de insulina, e gerou um número bastante inferior de testes abstratos. A maior desvantagem de utilizar a abordagem por simulação é o risco de não atingir a cobertura de todos os requisitos.

Os experimentos baseados na abordagem por espaço de estados e estrutural obtiveram comportamento semelhante ao experimento com o algoritmo DFS para a bomba de infusão de insulina completa, levando a resultados inesperados para esse modelo. Contudo, para os modelos de ECG e de bomba de infusão de insulina, a abordagem conseguiu gerar um número menor de testes abstratos, comparado ao experimento com DFS. Ainda assim, testes abstratos desnecessários foram gerados.

A maioria dos trabalhos identificados na RSL sugerem metodologias de modelagem e esse ponto pode ser considerado importante para conduzir a escolha pela abordagem. Para testadores que desejam utilizar MBT com o formalismo de CPN é recomendado considerar principalmente a complexidade do modelo, o tamanho do espaço de estados e o nível de cobertura de requisitos desejado para escolher a abordagem mais adequada.

Com esse estudo, foi possível identificar que a abordagem por espaço de estados é adequada para modelos com menor número de estados. A abordagem por simulação, ao contrário, é interessante para modelos complexos, com espaço de estados grande/infinito, porém ao custo de diminuir a cobertura dos requisitos. A abordagem estrutural mostrou ser outra opção adequada para modelos com espaço de estados grande/infinito, assistido por critérios de cobertura do grafo para melhorar a cobertura de requisitos, ao custo de implementações mais complexas e menos cobertura quando comparado com a abordagem por espaço de estados. A lacuna de pesquisa sobre a análise de abordagens para geração de testes abstratos com modelos de CPN foi preenchida, fornecendo insights para testadores que precisam aplicar o teste baseado em modelos.

Com a dificuldade em encontrar estudos que utilizam abordagem puramente estrutural, uma possível lacuna de pesquisa identificada nesse trabalho pode ser explorada por pesquisadores, uma vez que apenas um dos trabalhos identificados na RSL utiliza abordagem estrutural sem combiná-la com alguma outra abordagem disponível. Ainda assim, o artigo tem problemas de clareza e traz poucos detalhes sobre a condução do estudo.

Dados os resultados obtidos com o experimento com espaço de estados/estrutural em comparação com DFS, a abordagem estrutural tem potencial para contornar os problemas da abordagem por espaço de estados e mantendo seus benefícios, se aplicada de forma inteiramente estrutural. Além disso, abordagens baseadas em estrutura facilitam a geração de oráculos de testes e testes concretos, agilizando o processo de geração de testes como um

todo. Diante disso, pode ser considerado um tema com potencial para ser explorado.

Bibliografia

[1] Jensen K., & Kristensen L.M. (2015). Colored Petri Nets: A Graphical Language For Formal Modeling and Validation of Concurrent Systems. *Communications of the ACM*, *58*(6), 61-70. DOI: 10.1145/2663340.

[2] Murata T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, *77*(4),541–580. DOI: 10.1109/5.24143

[3] Li N., & Offutt J. (2017). Test Oracle Strategies for Model-Based Testing. *IEEE Transactions on Software Engineering*, *42*(4), 372-395. DOI: 10.1109/TSE.2016.2597136.

[4] Delamaro M. E., Maldonado J. C., Jino M. (2016). Introdução ao teste de software. LTC Exatas Didático, 2nd edição.

[5] Jensen K., & Kristensen L.M. (2009). *Coloured Petri Nets Modelling and Validation of Concurrent Systems*. Springer.

[6] Sobrinho A., Silva L. D., Perkusich A., Cunha P. ; Cordeiro T. & Lima, A. M. N. (2019). Formal modeling of biomedical signal acquisition systems: source of evidence for certification. Software & Systems Modeling, *18*, 1467-1485. DOI: 10.1007/s10270-017-0616-7.

[7] Zheng W., Liang C.,Wang R., & Kong W. (2014). Automated Test Approach Based on All Paths Covered Optimal Algorithm and Sequence Priority Selected Algorithm. In *IEEE Transactions on Intelligent Transportation Systems* (pp. 2551-2560). IEEE. DOI: 10.1109/TITS.2014.2320552.

[8] Zheng W. and Hu N., Automated Test Sequence Optimization Based on the Maze Algorithm and Ant Colony Algorithm,"*Int. J. Comput. Commun. Control*, vol. 10, pp. 593-606, 2015.

[9] Zhao X., Yang Z., & Lv J. (2016). Test generation approach based on Colored Petri Net of Mode Transition in on-board subsystem. In *35th Chinese Control Conference* (pp. 10134-10139). IEEE. DOI: 10.1109/ChiCC.2016.7554960.

[10] Zhang Y., Zhao X.Q., Zheng W., & Tang T. (2010). System Safety Property-oriented Test Sequences Generating Method Based On Model Checking. *WIT Transactions on The Built Environment*, *114*, 747-758. DOI: 10.2495/CR100681.

[11] Bernardino M., Rodrigues E.M., Zorzo A.F., & Marchezan L. (2017). Systematic mapping study on MBT: tools and models. *IET Software*, *11*(4),141-155. DOI: 10.1049/iet-sen.2015.0154.

[12] Uzun B., & Tekinerdogan B. (2018). Model-driven architecture based testing: A systematic literature review. *Information and Software Technology*, *102*, 30-48. DOI: 10.1016/j.infsof.2018.05.004.

[13] Kitchenham B., Charters S., Budgen D., Brereton P., Turner M., Linkman S., Jørgensen M., Mendes E., & Visaggio G. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. *EBSE Technical Report*.

[14] Wohlin C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *38th International Conference on Evaluation and Assessment in Software Engineering* (pp. 1-10). DOI: 10.1145/2601248.2601268.

[15] Costa T. F., Sobrinho A., Silva L. C., Silva L.D., & Perkusich A. (2019). A Coloured Petri Nets Reference Model of Insulin Infusion Pump Control Systems: Assisting the Certification Process. In *45th Annual Conference of the IEEE Industrial Electronics Society* (pp. 2871-2876). DOI: 10.1109/IECON.2019.8927111.

[16] Mertz L. (2018). Automated insulin delivery: Taking the guesswork out of diabetes management. *IEEE Pulse*, *9*(1), 8-9. DOI: 10.1109/MPUL.2017.2772685.

[17] International Consortium of Investigative Journalists. *Medical Devices Harm Patients Worldwide As Governments Fail On Safety*. Available on https://www.icij.org/investigations/implant-files/medical-devices-harmpatients-worldwide-as-governments-fail-on-safety/. 2018.

[18] Sun T., Ye X., & Liu J. (2011). A test sequence selection method for parallel software systems. In *Fourth International Symposium on Parallel Architectures, Algorithms and Programming* (pp. 163–167). IEEE. DOI: 10.1109/PAAP.2011.27.

[19] Sun T., Zhang T., & Guo X. (2018). Parallel software testing sequence generation method based on state pruning. In *IEEE International Conference on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications* (pp. 446–453). IEEE. DOI:10.1109/BDCloud.2018.00074.

[20] Sun T., Ye X., & Liu J. (2012). A test generation method based on model reduction for parallel software. In *13th International Conference on Parallel and Distributed Computing, Applications and Technologies* (pp. 777–782). IEEE. DOI: 10.1109/PDCAT.2012.144.

[21] Sun T., Wan X., Zhong W., Guo X., & Zhang T. (2020). Parallel software testing sequence generation method target at full covering tested behaviors. In *International Conference on Algorithms and Architectures for Parallel Processing* (pp. 59-67). Springer. DOI: 10.1007/978-3-030-38961-1_6.

[22] Cheng J., Zhao X., Liu J., & Zhang Y. (2019). Automated test generation based on colored Petri net and improved depth first search for train control system. In *Chinese Control Conference* (pp. 6761–6765). IEEE. DOI: 10.23919/ChiCC.2019.8866272.

[23] Lijie C., Tianshi Z., Chao S., & Hongyang H. (2018). Test case generation method based on colored Petri net for train control system. In *3rd International Conference on System Reliability and Safety* (pp. 1–5). IEEE. DOI: 10.1109/ICSRS.2018.8688869.

[24] Wu D., & Schnieder E. (2016). Scenario-based modeling of the on-board of a satellite-based train control system with colored Petri nets. *IEEE Transactions on Intelligent Transportation Systems*, *17*(11), 3045–3061. DOI: 10.1109/TITS.2016.2535418.

[25] Liu J., Ye X., & Li J. (2011). Colored Petri nets model based conformance test generation. In *IEEE Symposium on Computers and Communications* (pp. 967–970). IEEE. DOI: 10.1109/ISCC.2011.5983967.

[26] Ruan H., Wang L., Yang X., Dong L., & Li H. (2017). Openflow modeling based on CPN for evolution consideration and executable test case generation. In *41st Annual Computer Software and Applications Conference* (pp. 72–77). IEEE. DOI: 10.1109/COMPSAC.2017.183.

[27] Wang R., Kristensen L. M., & Stolz V. (2018). A tool for model-based software testing of distributed systems protocols using coloured Petri nets. In *International Conference on Verification and Evaluation of Computer and Communication Systems* (pp. 97-113). Springer. DOI: 10.1007/978-3-030-00359-3_7.

[28] Wang R., Kristensen L. M., Meling H., & Stolz V. (2019). Automated test case generation for the Paxos single-decree protocol using a coloured Petri net model. *Journal of Logical and Algebraic Methods in Programming*, *104*, 254-273. DOI: 10.1016/j.jlamp.2019.02.004.

[29] Cai L., Zhang J., & Liu Z. (2010). Generating test cases using colored Petri net. In *2nd International Symposium on Information Engineering and Electronic Commerce* (pp 1–5). IEEE. DOI: 10.1109/IEEC.2010.5533237.

[30] Farooq U., Lam C.P., & Li H. (2008). Towards automated test sequence generation. In *19th Australian Conference on Software Engineering* (pp. 441–450). IEEE. DOI: 10.1109/ASWEC.2008.4483233.

[31] Puspika B. N., Hendradjaya B., & Sunindyo W. D. (2015). Towards an automated test sequence generation for mobile application using colored Petri net. In *International Conference on Electrical Engineering and Informatics*, (pp 445–449). IEEE. DOI: 10.1109/ICEEI.2015.7352542.

[32] Silva B. C. F., Carvalho G., and Sampaio A. (2019). Cpn simulation-based test case generation from controlled natural-language requirements. *Science of Computer Programming*, *181*, 111-139. DOI: 10.1016/j.scico.2019.04.001.

[33] Jahan H., Rao S., & Liu D. (2016). Test case generation for bpel-based web service composition using colored Petri nets. In *International Conference on Progress in Informatics and Computing* (pp. 623–628). IEEE. DOI: 10.1109/PIC.2016.7949575.

[34] Zayaraz G., & Kalamegam P. (2016). Requirements driven test prioritisation approach for web service composition. *International Journal of Computer Applications in Technology*, *54*(4), p. 362. DOI: 10.1504/IJCAT.2016.080491.

[35] Kalamegam P., & Godandapani Z. (2012). Test sequences for web service composition using CPN model. *Computer Engineering and Intelligent Systems*, *3*(6), pp. 32–41.

[36] Wang Y., & Chen P. (2013). Test case generation of web service composition: An approach based on the color Petri net. *Applied Mechanics and Materials*, *336-338*, pp. 2063–2070. DOI:10.4028/www.scientific.net/AMM.336-338.2063.

[37] Cai L. (2011). A Business Process Testing Sequence Generation Approach Based on Test Cases Composition. *First ACIS/JNU International Conference on Computers, Networks, Systems, and Industrial Engineering*, (pp. 179-185) DOI 10.1109/CNSI.2011.12.

[38] Dong L., Li H., & Zhang S. (2013). Workflow-based Modeling of Web Application and Automatically Generating Test Sequences*. *13th International Conference on Quality Software* (pp. 378-381) IEEE. DOI 10.1109/QSIC.2013.37.

[39] Wu D., & Krause J. (2013). Model-based Test Generation Techniques Verifying the On-board Module of a Satellite-based Train Control System Model. IEEE.

[40] Zhao X., Zhang Y., Zheng W., Tang T., & Mu R. (2010). Modelling and design of the formal approach for generating test sequences of ETCS level 2 based on the CPN. *WIT Transactions on The Built Environment, Vol 114* (pp. 723-734) WIT Press. DOI 10.2495/CR100661.

[41] Neto A. D., Subramanyan R., Vieira M., & Travassos G. H. (2007). A Survey on Model-based Testing Approaches: A Systematic Review *ACM International Conference on Automated Software Engineering-ASE* (pp 31-36) DOI 10.1145/1353673.1353681.

[42] Haser F., Ferderer M., & Breu R. (2014). Software Paradigms, Assessment Types and Non-Functional Requirements in Model-Based Integration Testing: A Systematic Literature Review. *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering* (pp. 1-10). DOI 10.1145/ 2601248.2601257.

[43] Shirole M., & Kumar R. (2013). UML Behavioral Model Based Test Case Generation: A Survey. *ACM SIGSOFT Software Engineering Notes* (pp. 1-13). DOI: 10.1145/2492248.2492274.

[44] Petry K. L., Oliveira E.J., & Zorzob A. F. (2020). Model-Based Testing of Software Product Lines: Mapping Study and Research Roadmap. *Journal of Systems and Software* DOI 10.1016/j.jss.2020.110608.

[45] Ahmad T.,Iqbal J.,Ashraf A., Truscan D., & Porres I. (2019). Model-based testing using UML activity diagrams: A systematic mapping study. *Computer Science Review 33* (pp. 98-112). DOI 10.1016/j.cosrev.2019.07.001.

[46] Gurbuz, H. G., & Tekinerdogan, B. (2018). Model-based testing for software safety: a systematic mapping study *Software Quality Journal* (pp. 1327-1372). DOI 10.1007/s11219-017-9386-2.

[47] Shafique, M., & Labiche, Y. (2015). A systematic review of state-based test tools. *International Journal on Software Tools for Technology Transfer* (pp.59-76). DOI 10.1007/s10009-013-0291-0.

[48] Sabbaghi A., & Keyvanpour M. R. (2017). State-based models in model-based testing: A systematic review. *2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)* (pp. 942-948). DOI 10.1109/KBEI.2017.8324934.

[49] Saeed A., Hamid S. H. A., & Mustafa M. B. (2016). The experimental applications of search-based techniques for model-based testing: Taxonomy and systematic literature review. *Applied Soft Computing* (pp. 1094 - 1117). DOI 10.1016/j.asoc.2016.08.030.

[50] Siavashi F., & Truscan D. (2015). Environment Modeling in Model-based Testing: Concepts, Prospects and Research Challenges: A Systematic Literature Review. *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering* (pp. 31-36). DOI 10.1145/2745802.2745830.

[51] Paiva S. L. d. C., & Simão A.D.S. (2015).A Systematic Mapping Study on Test Generation from Input/Output Transition Systems. *2015 41st Euromicro Conference on Software Engineering and Advanced Applications* (pp. 333-340). DOI 10.1109/SEAA.2015.66.

[52] Rafi D.M.,Moses K.R., Petersen K. & Mäntylä, M.V.(2012). Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. *7th International Workshop on Automation of Software Test, AST 2012 - Proceedings* (pp. 36-42). DOI 10.1109/IWAST.2012.6228988.

[53] Saxena R., & Singhal A. (2017). A critical review of mutation testing technique and hurdles. 2017 International Conference on Computing, Communication and Automation (ICCCA) (pp 887-892). DOI 10.1109/CCAA.2017.8229932.

[54] Veanes M., Campbell C., Grieskamp W., Schulte W., Tillmann N., & Nachmanson L.(2008). Model-Based Testing of Object-Oriented Reactive Systems with Spec Explorer *Formal Methods and Testing: An Outcome of the FORTEST Network, Revised Selected Papers* (pp. 39-76). DOI 10.1007/978-3-540-78917-8_2.

[55] Bashir M. F., & Banuri S. H. K. (2008). Automated model based software Test Data Generation System. *4th International Conference on Emerging Technologies* (pp. 275-279). DOI 10.1109/ICET.2008.4777514.

[56] Myers G. J. (2012). The Art of Software Testing *John Wiley & Sons, Ltd* DOI 10.1002/9781119202486.

Anexo 1: Siglas

CPN - Coloured Petri Nets.

RTL - Revisão Terciária da Literatura.

RSL - Revisão sistemática da literatura.

SML - Standard ML. Linguagem de paradigma funcional.

CPN/ML - Linguagem de paradigma funcional para modelos de CPN que deriva do padrão SML.

ECG - Eletrocardiografia.

MBT - Model Based Testing, termo em inglês para Testes Baseados em Modelos

APCO - All Paths Covered Optimal. Algoritmo utilizado para geração de testes abstratos

SPS - Sequence Priority Selected. Algoritmo utilizado para priorização de testes abstratos.

CPN/Tools - Ferramenta disponível para criação e manipulação de modelos de CPN.

DFS - Depht first search. Algoritmo tradicional de busca em profundidade utilizado para percorrer grafos.